CMSC 661, Fall 1999, Principal of Database Systems

Final Report, Dec. 7

Team #3
Honglin Su, hosu@cs.umbc.edu
Shilang Tang, stang2@cs.umbc.edu
Zhongli Ding, zding1@cs.umbc.edu

Abstract:

As our project's final report, as well as accumulated report, it gives the progress and update of the project. We finished our database design, and all GUI design. We made a few modifications for the relational schema we proposed in the previous reports. In the last section, we also point out the limitation of our implementation and possible improvement. A detailed user manual is given as appendix of the manual.

Table of Contents

Abstract:.... 1
1. Introduction to the Database Application.... 2
2. Database Design.... 2
2.1 The Assumption made to run the application.... 2
2.2 Tasks to implement.... 3
2.3 E-R Diagram.... 4
2.4. Attributes for each entity and relationship.... 7
2.5 Relation Schema.... 10
2.6 Issues.... 13
3. Implementation.... 14
3.1 SQL statement.... 14
3.2 Issues to highlight.... 22
4. Limitation and possible improvements.... 24
Appendix.... 26
A.1 List of Source Codes.... 26
A.2 User Manual.... 28

1. Introduction to the Database Application

With the development of the Department of Computer Science and Electrical Engineering (CSEE) at University of Maryland, Baltimore County, the department want to automate the management of graduate students, including those who are seeking admission to the department, currently enrolled students, and those successfully graduated or unfortunately dismissed.

This project is developed for the Graduate Students Office (GSO) of CSEE department at UMBC. Since CSEE is a large department and so has a large number of students, and every student occupies a lot of information should be maintained by the GSO staff, it is painful to record all these information in paper and manage them by human resource. And also, it is inconvenient for a faculty if s/he want to know some kind of information about some specific students. But by using computer technology, we can overcome all these drawbacks. Thus, Our goal is to develop a student information management system via Java, database and web technology, which can make our users handle and check all kinds of student information easily and efficiently, most importantly the application can run across different system platforms. Here, our users of this database application are going to be staff from the GSO and faculty in the CSEE department. For example, a faculty can access a student's information easily if he has a network computer in his office and get an account in our system. We Suppose most of the modifications to the information in the database will be done by GSO staff, while faculty will be accessing the database mostly in read-only mode.

In detail, there are 4 basic student types: Students seeking admission into a graduate program, Students that enrolled in a program, Students that graduated, Students that are dismissed from the program. Our users should can query or update a specific student's information directly via providing the student's SSN, and also, s/he should can query or update database via student type. Basically, our system should can handle the following major types of events: events happened to an applicant students, events happened to an enrolled students such as the change or query of the student's academic information, and events such as a student changes her/his status, from one type of student to another kind of student, etc.

In a word, our purpose is to provide the management functions as completely as possible. And also, we will provide a friendly user interface. By using our system, the staff and faculty can handle the information easily.

2. Database Design

2.1 The Assumption made to run the application

The CSEE department has a networked computing environment, every computer in the department including PCs, Macs, Unix workstations or servers, whether they are running

Windows (95/98/NT), Solaris, or Irix, they all have Java virtual machine installed, suppose they all have Java 2 ready, this is very fundamental assumption. Our application will use Java, JDBC, and Oracle, and they application can run every Java platforms.

We will also take advantage of the advanced features of the Oracle 8 DBMS. In order to make the design more efficient, we will use user-defined data types. For example, we will define a data type Address_Type which has several attributes such as street, number, city, state, zip, and country. We will also use variable array to represent many-one relation, such as committee information, which usually has a maximum number of members such as five or six, then we can define an array of with maximum six members, each member has two attributes, i.e. name and role.

2.2 Tasks to implement

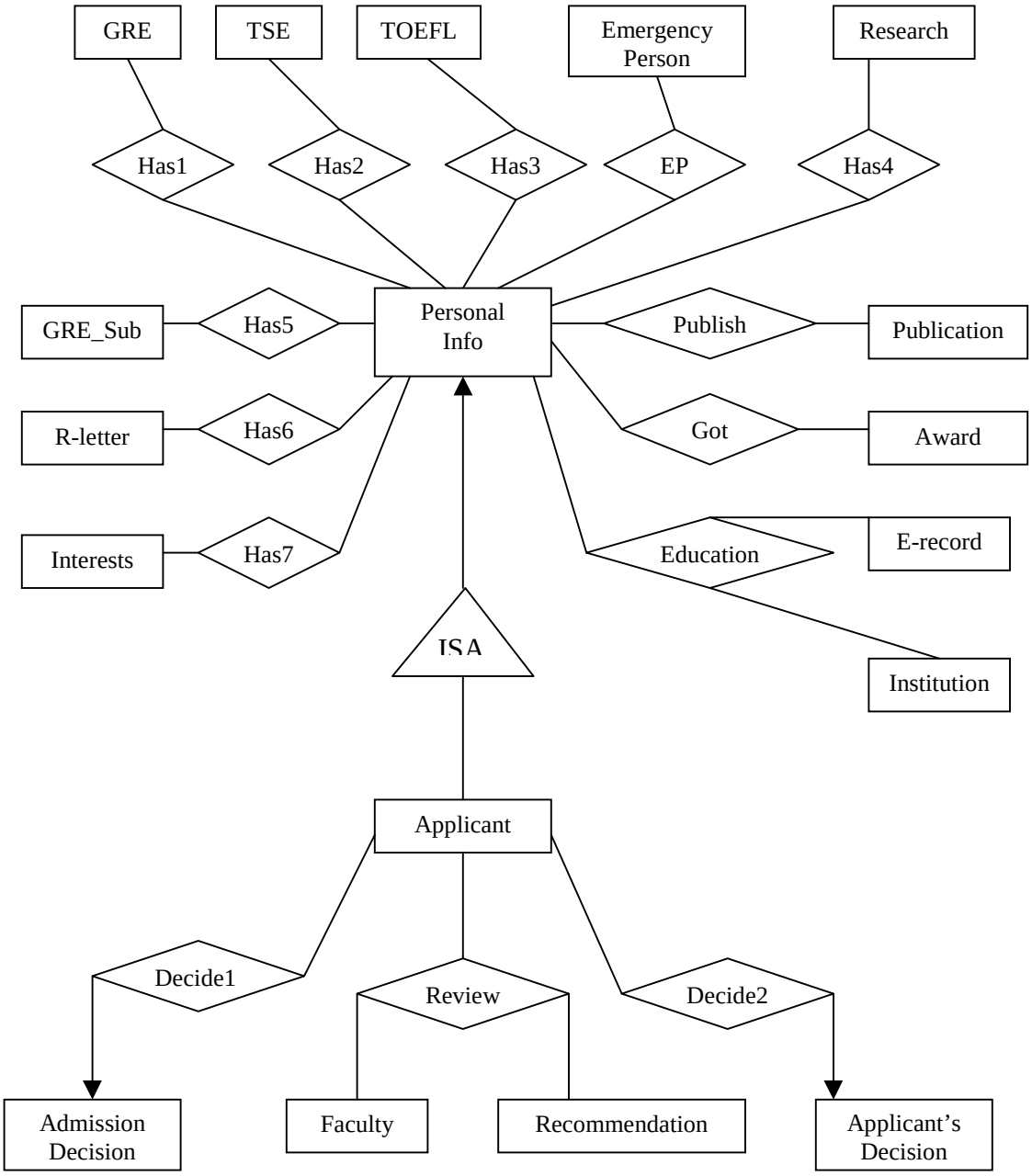
We are considering to realize the following tasks, which handle the corresponding events.

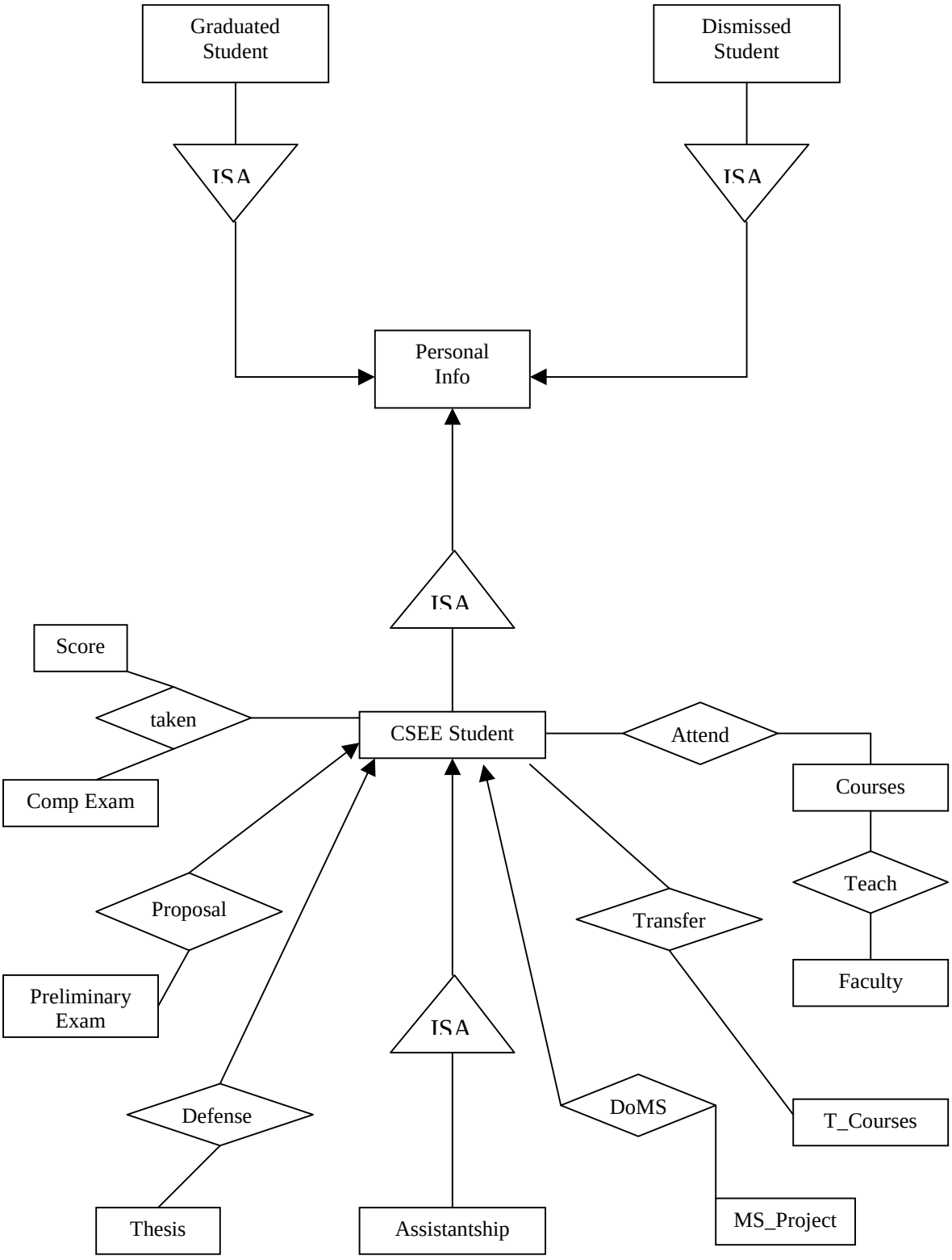
1. Receive a new application, add new personal information into the database;
2. Update personal information when supporting materials are arriving;
3. When all application materials are received, update the status;
4. Application is sent to reviewer, grant appropriate user rights to the reviewer;
5. Reviewer makes a recommendation for the applicant, update the reviewer's table;
6. Application is received from reviewer, revoke the user rights to access the application;
7. Application is sent to admissions committee, grant appropriate user rights to admission committee;
8. Application is received from admissions committee, revoke the user rights to access the application;
9. Committee makes a decision, update the applicant status;
10. Letters are sent (departmental, graduate school, international office), update the dates respectively;
11. Applicant replies, update the applicant information;
12. Admissions committee approves defer request if the defer request is received by the department, update applicant status.
13. Admitted student enrolls in a program, update the student's information. A new row is inserted in the student table.
14. Student change status, update the student information;

15. Student takes classes, update the student and courses relation information;
16. Student changes academic/research advisors, update the student information;
17. Student takes comprehensive exams, add an entry to comprehensive table;
18. Student forms his/her advisory, examination, or defense committees, update the committee information;
19. Student takes Ph.D. preliminary exam, add an entry to the Ph.D. preliminary table;
20. Student defends thesis, update the thesis table;
21. Student completes M.S. project, update the MS. Project table;
22. Student is admitted into Ph.D. candidacy, add an entry to Ph.D. candidacy table;
23. Student receives an award, add an entry to award table;
24. Student is awarded an assistantship, add an entry to assistantship table;
25. Student is dismissed from his/her assistantship, update assistantship table and student status;
26. Student is dismissed from the program, add an entry to dismissed student table;
27. Student graduates, add an entry to graduated student table.

2.3 E-R Diagram

In this diagram, each entities and associated relationships are given. For attributes, they are listed in 2.4.


GRE
TSE
TOEFL
Emergency Person
Research
Has1
Has2
Has3
EP
Has4
GRE_Sub
Has5
Personal Info
Publish
Publication
R-letter
Has6
Got
Award
Interests
Has7
Education
E-record
ISA
Institution
Applicant
Decide1
Review
Decide2
Admission Decision
Faculty
Recommendation
Applicant's Decision


Graduated Student
Dismissed Student
ISA
ISA
Personal Info
ISA
Score
taken
Comp Exam
CSEE Student
Attend
Courses
Teach
Faculty
Proposal
Preliminary Exam
Transfer
T_Courses
ISA
Defense
Thesis
Assistantship
DoMS
MS_Project

2.4. Attributes for each entity and relationship

Here we list attributes for each entity and relationship, we also give explanation for the non-obvious attributes, entities, and relationships.

Personal_Info (SSN, name, sex, bdate, ethnic, citizenship, language, home_address, work_address, home_phone, work_phone, email)
Object type, Address, has the following attributes: street, number, city, state, zip, country.

Emergency_Person (name, relationship, address, phone, email, primary)
Object type, Address, has the following attributes: street, number, city, state, zip, country.

EP(SSN, name)

Institution (NO#, name, address)
Object type, Address, has the following attributes: street, number, city, state, zip, country.

Education (SSN, degree, gradudate_date, Institution_No, major, minor, GPA, GPA_scale, received_transcript, received_degree_certificate)

E-record (degree, major, graduate_date, minor, GPA, GPA_scale, received_transcript, received_degree_certificate)

R-letter (item#, name, title, address, received_date, text)

Has6 (SSN, item#)

GRE (test_date, verbal, quantitative, analytical)

Has1 (SSN, test_date, verbal, quantitative, analytical, waived)

GRE_Sub (test_date, subject, score)

Has5 (SSN, Test_date, Subject, score, waived)

TOEFL (test_date, listening, writing, comprehension)

Has3 (SSN, test_date, listening, writing, comprepension, waived)

TSE (test_date, score)

Has2 (SSN, test_date, score, waived)

Research (keywords)

Has4 (SSN, keywords)

Interests (Interest)

Has7 (SSN, Interest)

Publication (item#, title, authors, type, month, year, citation)

Publish (SSN, item#)

Award (item#, name, date, description)

Got (SSN, item#)

Applicant(SSN, received_date, program, degree_sought, semester, semester_year, financial_support_requested, financial_support_source, status, last_update_date, dept_send_date, grad_send_date, intl_send_date, notes)

Faculty (Name, SSN)

Recommendation(item#, review_date, recommendation, matching_faculty, required_courses, comments)

Review(SSN, item#, Name)

Admission_Decision (item#, admission_date, decision, deny_reason, provisions, assigned advisor, waived_tests, assistantship)
assistantship is an object type, which has (awarded, type, amount, duration).

Decide1 (SSN, item#)

Applicant_Decision (item#, decision_date, decision, expect_arrive_date, defer_requent, notes)

Decide2 (SSN, item#)

Student (SSN, program, admitted_date, degree_sought, academci_advisor, research_advisor, status, phd_candicacy_date, expected_grad_date, xerox_no, last_update_date)

Comp_Exam (subjects, date)

Score(item#, score, grade)

Taken(SSN, item#, subjects, date)

Preliminary_Exam (item#, test_date, title, abstract, grade, committee)

committee is an array of object type (name, role)

Proposal (SSN, item#)

Thesis (item#, degree, title, abstract, defended_date, result, committee)

Defense(SSN, item#)

MS_project (item#, approved_date, title, abstract, committee)

DoMS(SSN, item#)

Courses (course_number, semester, year, course_name)

Teach (name, course_number, semester, year)

Attend (student_SSN, course_number, semester, year, grade)

T_Courses (Item#, courses_number, transfer_institution, grade)

Transfer (SSN, item#)

Graduate_Assistant (Assistant_SSN, start_date, assistant_type, step, level, duration, account, remission_credits, stipend, health benefit, supervisor, evaluation, duty, office, phone, TA_test_score, TA_assignment, dismissed_date, dismissed_reason)
TA_assignment is an array of an object type (course, section, level, instructor, evaluation)

Graduated_student (student_SSN, degree, awarded_date, employment)
employment is an objec type (company, start_date)

Dismissed_student (student_SSN, dismissal_date, reasons)

2.5 Relation Schema

We map the E-R diagram to relational schema according to 3NF. This is a revised version of report #1.

- Users (username, ssn, password, status)

The table stores user information for management.

- Superstaff (username, password)

This table stores users, who has the highest privilege to manage other users.

The above two tables are for administration usage.

- Person (SSN, name, sex, bdate, ethnic, citizenship, language, home_address, work_address, home_phone, work_phone, email)

We define an object type Address, which has the following attributes: street, city, state, zip, country.
Person Information is the basic table for each student that seeks admission or is /was enrolled in a program. SSN is the primary key.

- Emergency_Contact_Person (SSN, name, relationship, address, phone, email, primary)

This table record the student's emergency contact persons, the attribute primary indicates that whether this person is the primary emergency contact person. It borrows SSN from Personal_Info, both SSN and name are the key.

- Institution (No, name, address)

The Institution has the No. as the key, which is similar to ETS number for each institution.

- Education (SSN, degree, gradudate_date, Institution_No, major, minor, GPA, GPA_scale, received_transcript, received_degree_certificate)

Use SSN, degree, and graduate_date as the key. It also borrow the Institution_No from the table Institution.

- Recommendation (SSN, name, title, address, received_date)

- GRE (SSN, test_date, verbal, quantitative, analytical, waived)

- GRE_Sub (SSN, test_date, subject, score, waived)

- TOEFL (SSN, test_date, listening, writing, comprehension, waived)

- TSE (SSN, test_date, score, waived)

- Research_keyword (name)

This table stores keyword dictionary.

- Research (SSN, keywords, interests)

We define a VARRAY type for keywords;

- Publication (SSN, title, pub_type, authors, pub_date, citation)

We define a VARRAY type for authors;

- Award (SSN, award_name, award_date, description)

- Applicant(SSN, received_date, program, degree_sought, semester, semester_year, financial_support_requested, financial_support_source, status, last_update_date, dept_send_date, grad_send_date, intl_send_date, notes)

- Faculty (Name)

- Reviewer(Faculty_name, applicant_SSN, review_date, recommendation, matching_faculty, required_courses, comments)

- Deny_reason (name)
  This table stores deny reason dictionary.

- Prerequest_course (cno, name)
  This table stores prerequisite course dictionary.

- Admission_Decision (applicant_SSN, admission_date, decision, deny_reason, provisions, assigned advisor, waived_tests, assistantship, requiredcourse)

We define an object type for assistantship, which has (awarded, type, amount, duration).
We also define a VARRAY type for prerequisite courses.

- Applicant_Decision (applicant_SSN, decision_date, decision, expect_arrive_date, defer_requent, notes)

- Student_Info (SSN, program, admitted_date, degree_sought, academci_advisor, research_advisor, status, phd_candicacy_date, expected_grad_date, xerox_no, last_update_date)

- Comp_Exam (Student_SSN, subjects, result)
  We define a VARRAY of an object type (subject, test_date, score, grade) for subjects.

- Phd_Preliminary (Student_SSN, test_date, title, abstract, grade, committee)
  We define a VARRAY of object type (name, role) for committee.

- Thesis (student_SSN, defended_date, degree, title, abstract, result, committee)

- MS_project (student_SSN, approved_date, title, abstract, committee)

- UMBC_Course (cno, cname)
  This table stores course dictionary.

- Course (course_number, semester, year, course_name, instructor)
  Course schedule for each semester.

- Transfer_Course (course_number, SSN, institution, grade)

- Attended_courses(student_SSN, course_number, semester, year, grade, transfer_course, transfer_institution)

- Graduate_Assistant (Assistant_SSN, start_date, assistant_type, step, level, duration, account, remission_credits, stipend, health benefit, supervisor, evaluation, duty, office, phone, TA_test_score, TA_assignment, dismissed_date, dismissed_reason)

TA_assignment is an array of an object type (course, section, level, instructor, evaluation)

- Graduated_student (student_SSN, degree, awarded_date, company, startdate)

- Dismissed_student (student_SSN, dismissal_date, reasons)

2.6 Issues

1. GUI design is a time-consuming task. We plan to utilize Java JFC (Swing) in our application, the first we want to implement the independent Java application, which can run any Java 2 platform.

2. Choose a proper architecture, a 2-tier or 3-tier. For a 2-tier model, the first tier is machine which runs the application, and the second tier is the Oracle 8 database, and a 3-tier model, it has a Web server which will direct any client request to the back Oracle 8 database. Currently, we are using 2-tier model.

3. Implementation

We are using Oracle Jdeveloper 2.0 as our user interface design environment. In order to make our database design clear and concise, we try to make full use of the advanced features of the Oracle 8I, e.g. user-defined data type, varying arrays. However, the more we use the advanced features, the more workload for the user interface design.

3.1 SQL statement

```
create table users (
    username      varchar2(15)      primary key,
    ssn           varchar2(9),
    pwd           varchar2(20),
    status        char
);

create table superstaff (
    uname         varchar2(15)      primary key,
    upswd         varchar2(15)
);

create type address_type as object (
    street        varchar2(50),
    city          varchar2(20),
    state         varchar2(20),
    zip           varchar2(10),
    country       varchar2(30)
);
/
create table person (
    ssn           varchar2(9)       primary key,
    name          varchar2(30)      not null,
    sex           char,
    bdate         date              not null,
    ethnic        varchar2(10),
    citizen       varchar2(30)      not null,
    language      varchar2(20)      not null,
    homeaddr      address_type,
    workaddr      address_type,
    hphone        varchar2(10),
    wphone        varchar2(10),
    email         varchar2(30)      not null
);

create table ECPerson (
    forssn        varchar2(9),
    name          varchar2(30)      not null,
    relation      varchar2(20)      not null,
    address       address_type,
    phone         varchar2(10)      not null,
    email         varchar2(30)      not null,
    pricontact    char              default 'y',
    primary key   (forssn, name),
    foreign key   (forssn) references person(ssn)
                  on delete cascade
```

```
);

create table institution (
    ino           varchar2(6)       primary key,
    name          varchar2(30),
    address       address_type
);

create table education (
    forssn        varchar2(9),
    degree        varchar2(10),
    graddate      date,
    instno        varchar2(6),
    major         varchar2(20),
    minor         varchar2(20),
    gpa           varchar2(6),
    gpa_scale     varchar2(6),
    rcv_tran      char              default 'n',
    rcv_cert      char              default 'n',
    primary key   (forssn, degree, graddate),
    foreign key   (forssn) references person(ssn)
                  on delete cascade,
    foreign key   (instno) references institution(ino)
);

create table recommendation (
    forssn        varchar2(9),
    name          varchar2(30),
    title         varchar2(10),
    address       address_type,
    recv_date     date,
    primary key   (forssn, name),
    foreign key   (forssn) references person(ssn)
                  on delete cascade
);

create table gre (
    forssn        varchar2(9),
    testdate      date,
    verbal        number            not null,
    quant         number            not null,
    anal          number            not null,
    waived        char              default 'n',
    primary key   (forssn, testdate),
    foreign key   (forssn) references person(ssn)
                  on delete cascade
);

create table gresub (
    forssn        varchar2(9),
    testdate      date,
    subject       varchar2(20),
    score         number            not null,
    waived        char              default 'y',
    primary key   (forssn, testdate, subject),
    foreign key   (forssn) references person(ssn)
```

```
                on delete cascade
);

create table toefl (
    forssn        varchar2(9),
    testdate      date,
    listen        number            not null,
    write         number            not null,
    comp          number            not null,
    waived        char              default 'n',
    primary key   (forssn, testdate),
    foreign key   (forssn) references person(ssn)
                  on delete cascade
);

create table tse (
    forssn        varchar2(9),
    testdate      date,
    score         number            not null,
    waived        char              default 'y',
    primary key   (forssn, testdate),
    foreign key   (forssn) references person(ssn)
                  on delete cascade
);

create table research_keyword (
    name          varchar2(15)      primary key
);

create type author_array as varray(4) of varchar2(30);
/

create type keywords_array as varray(4) of varchar2(20);
/

create table publication (
    forssn        varchar2(9),
    ptype         varchar2(20),
    title         varchar2(30),
    authors       author_array,
    pdate         varchar2(12),
    citation      varchar2(80),
    primary key   (forssn, title),
    foreign key   (forssn) references person (ssn)
                  on delete cascade
);

create table research (
    forssn        varchar2(9)       primary key,
    keywords      keywords_array,
    interests     varchar2(256),
    foreign key   (forssn) references person (ssn)
                  on delete cascade
);

create table award (
    forssn        varchar2(9),
```

```
    aname         varchar2(30),
    adate         date              not null,
    description   varchar2(80),
    primary key   (forssn, aname, adate),
    foreign key   (forssn) references person (ssn)
                  on delete cascade
);

create table applicant (
    forssn        varchar2(9)       primary key,
    rcvdate       date,
    program       char(4)           not null,
    degree        varchar2(10)      not null,
    semester      varchar2(10)      not null,
    semyear       number            not null,
    finrequest    char              default 'y',
    finsource     varchar2(80),
    status        varchar2(12)      not null,
    lastdate      date              not null,
    deptdate      date,             /* date sent to department */
    graddate      date,             /* date sent to grad school */
    intldate      date,             /* date sent to intl office */
    notes         varchar2(80),
    foreign key   (forssn) references person (ssn)
                  on delete cascade
);

create table faculty (
    name          varchar2(30)      primary key
);

create table reviewer (
    fname         varchar2(30),
    forssn        varchar2(9),
    revdate       date,
    recomm        varchar2(80),
    match         varchar2(30),     /* matching faculty */
    reqcourses    varchar2(80),
    comments      varchar2(80),
    primary key   (fname, forssn),
    foreign key   (fname) references faculty(name),
    foreign key   (match) references faculty(name),
    foreign key   (forssn) references person(forssn)
                  on delete cascade
);

create type assist_type as object (
    awarded       char,
    atype         varchar2(10),
    amount        number,
    begdate       date,
    enddate       date
);
/
create table deny_reason (
    name          varchar2(30)      primary key
);
```

```
create table prerequest (     /* undergraduate courses required */
    cno           varchar2(8)       primary key,
    name          varchar2(30)
);

create type cno_array as varray(3) of varchar2(8);
/

create table admdecision (
    forssn        varchar2(9)       primary key,
    admdate       date              not null,
    decision      varchar2(20)      not null,
    denyreason    varchar2(30),
    provisions    varchar2(30),
    advisor       varchar2(30),
    waivedtests   varchar2(30),
    reqcno        cno_array,
    assist        assist_type,
    foreign key   (forssn) references person (forssn)
                  on delete cascade
);

create table appdecision (
    forssn        varchar2(9)       primary key,
    decdate       date              not null,
    decision      varchar2(20)      not null,
    expdate       date,             /* expect arrive date */
    deferreq      char              default 'n',
    notes         varchar2(80),
    foreign key   (forssn) references person(forssn)
                  on delete cascade
);

create table student (
    forssn        varchar2(9)       primary key,
    program       char(4)           not null,
    admdate       date,
    degree        varchar2(10)      not null,
    acadvisor     varchar2(30),
    resadvisor    varchar2(30),
    status        varchar2(10),
    phdcandate    date,             /* Ph.D. candiate date */
    expgraddate   date,             /* expect graduation date */
    xeroxno       number(5),
    lastdate      date,
    foreign key   (forssn) references person (forssn)
                  on delete cascade,
    foreign key   (acadvisor) references faculty (name),
    foreign key   (resadvisor) references faculty (name)
);

create type subject_type as object (
    subject       varchar2(30),
    testdate      date,
    score         number,
    grade         varchar2(10)
```

```
);
/
create type subject_array as varray(8) of subject_type;
/
create table compexam (
    forssn        varchar2(9)       primary key,
    subjects      subject_array,
    result        varchar2(10),
    foreign key   (forssn) references student(forssn)
                  on delete cascade
);

create type committee_array as varray(6) of varchar2(30);
/
create table phdprelim (
    forssn        varchar2(9),
    testdate      date,
    title         varchar2(80)            not null,
    abstract      varchar2(256),
    grade         varchar2(10),
    committee     committee_array,
    primary key   (forssn, testdate),
    foreign key   (forssn) references student(forssn)
                  on delete cascade
);

create table thesis (
    forssn        varchar2(9),
    defdate       date,
    degree        varchar2(10)            not null,
    title         varchar2(80)            not null,
    abstract      varchar2(256),
    result        varchar2(10),
    committee     committee_array,
    primary key   (forssn, defdate),
    foreign key   (forssn) references student(forssn)
                  on delete cascade
);

create table msproject (
    forssn        varchar2(9),
    appvdate      date,
    title         varchar2(80)      not null,
    abstract      varchar2(256),
    committee     committee_array,
    primary key   (forssn, appvdate),
    foreign key   (forssn) references student(forssn)
                  on delete cascade
);

create table umbc_courses (
    cno           varchar2(8)       primary key,
    cname         varchar2(80)
);

create table courses (
    cno           varchar2(8),
```

```
    semester      varchar2(6),
    year          number(4),
    cname         varchar2(80)      not null,
    instructor    varchar2(30),
    primary key   (cno, semester, year),
    foreign key   (cno) references umbc_courses(cno)
);

create table transcourse (
    cno           varchar2(7),
    semester      varchar2(6),
    year          number(4),
    forssn        varchar2(9),
    instno        varchar2(6),
    grade         char,
    primary key   (cno, semester, year, forssn),
    foreign key   (cno, semester, year)
                  references courses (cno, semester, year),
    foreign key   (forssn) references student(forssn)
                  on delete cascade,
    foreign key   (instno) references institution (ino)
);

create table attcourses (
    forssn        varchar2(9),
    cno           varchar2(7),
    semester      varchar2(6),
    year          number(4),
    grade         char,
    primary key   (forssn, cno, semester, year),
    foreign key   (forssn) references student (forssn)
                  on delete cascade,
    foreign key   (cno, semester, year)
                  references courses (cno, semester, year)
);

create type ta_type as object (
    cno           varchar2(7),
    semester      varchar2(6),
    year          number(4),
    section       number(4),
    talevel       char,
    instructor    varchar2(30),
    evaluation    varchar2(10)
);
/
create type ta_array as varray(10) of ta_type;
/
create table gradassist (
    forssn        varchar2(9),
    startdate     date,
    asstype       varchar2(10),
    step          number(1),
    enddate       date              not null,
    account       varchar2(20)      not null,
    remiscredit   number(2)         not null,   /* remission credits */
    stipend       number(7, 2),
```

```
    health        varchar2(20),
    supervisor    varchar2(30),
    evaluation    varchar2(80),
    duty          varchar2(80),
    office        varchar2(20),
    phone         varchar2(10),
    tatestscore   char(2),
    assignment    ta_array,
    disdate       date,             /* dismiss date */
    disreason     varchar2(80),
    primary key   (forssn, startdate),
    foreign key   (forssn) references student(forssn)
                  on delete cascade
);

create table graduation (
    forssn        varchar2(9)       primary key,
    degree        varchar2(10)      not null,
    awdate        date              not null,
    company       varchar2(80),
    startdate     date,
    foreign key   (forssn) references student(forssn)
                  on delete cascade
);

create table dismission (
    forssn        varchar2(9)       primary key,
    disdate       date              not null,
    reasons       varchar2(80),
    foreign key   (forssn) references student (forssn)
                  on delete cascade
);

create index person_idx on person(ssn);

create index applicant_idx on applicant(forssn);

create index student_idx on student(forssn);
```

3.2 Issues to highlight

1. JDBC connection

   Since building a JDBC connection is a time-consuming procedure, we make only one connection for the whole operation procedure of each staff or faculty. When the staff or faculty switches different menu, which is corresponding to the Oracle table, we keep the JDBC connection, however, for Statement or ResultSet, we may create it or close it upon any requirement. When the user finishes all operation, the JDBC connection will be closed.

2. How to keep data integration?

   In most cases, we will use primary key constraint and referential constraint to keep data integrity. We also specify the value domain, which give the user choices, thus to make sure that user can not have any typo error. e.g. For research keywords, only the keywords stored in the keyword table can be chosen. For faculty members, only if they are already in the faculty table, can they be assigned to review applications. we also give the staff an administration frame to add new keyword, add new faculty, new users and so on.

3. How to implement friendly user interface?

   Ideally, we want to offer user a very friendly interface. When the user input any value to a field, if there is any error, we can point it out the right location and what the error is. e.g. To input the SSN, which is 9-digit number, if a user types less than 9 digits or other characters, this field would be marked up.

   One difficulty point is to dynamically show the query results, if there is more than one rows, and some columns are user-defined object type.

4. How to manipulate the object data?

   In our schema design, we use quite a few user-defined object type. It is easy to insert data into the table, however, how can we extract data out of the database? e.g. We define an address type as following.

```
create type address_type as object (
    street          varchar2(50),
    city            varchar2(20),
    state           varchar2(20),
    zip             varchar2(10),
    country         varchar2(30)
);
```

   There is a simple table as following:

```
create table person(
    name            varchar2(30)      primary key,
    address         address_type
);
```

When we do a query, we want to retrieve the city of the specified name. First, we do a query to get the whole address_type, then get the information that we want. So, in the Java class, we have the following code.

```
// Query the person table
Statement stmt = conn.createStatement();
ResultSet rset = stmt.executeQuery
               ("select address from person where name='"
                 + name + "'");
// name is the user input value
if (rset.next()) {
   Struct myStruct = (Struct) rset.getObject(1);

    Object[] allattr = (Object[])myStruct.getAttributes();

// since we already know the order of column in the address_type,
// so allattr[1] represent the value of the city.

    if (allattr[1] != null)
       System.out.println("The city when " + name + " lives:: " +
       allattr[1].toString());
```

If we want to retrieve other object data, we can do it in a similar way. For varying array, the difference is that we use the method getArray(#) of ResultSet.

5. Using Oracle stored procedure or Java class?

   To maintain integrity of the database and perform appropriate actions on database update, shall we use Oracle supplied trigger or Java class? At this stage, we only use Oracle supplied cascade update or delete trigger, most of database updates are realized by Java class through JDBC.

   For example, to decide whether the applicant's status is complete, there is 3 ways to do this. First, use Java code to search all corresponding tables, check relevant attributes, and get the answer. The second way is to use Oracle stored procedure, let the user call this procedure or make it as a trigger, invoked by a special event. The third way is to do it manually, let the user manually set the application's update status. In this stage, we have implemented the third way, and are trying to implement the first or the second based on the progress of the project.

4. Limitation and possible improvements

We implement this project by pure Java, JDBC, and take advantage of the advanced features of the Oracle 8 DBMS. The GSO management system can run on every platforms which have Java virtual machine (JDK 1.1 and above). However, due to different system environment, the display fonts are a little different among those system. We have tested our application on Sun Solaris 2.6 (JDK 1.2), SGI Irix 6.5 (JDK 1.1), Windows NT (JDK 1.2) and Win95 (JDK 1.2). The application fits exactly on Windows, it looks not very beautiful on SGI. We need to change the font parameters to fix this problem.

At present, we focus on how to keep data integration, how to provide a friendly user interface, and how to improve its performance. We suppose that the GSO management system is to facilitate the management of CSEE graduate students and applicants. It could be more intelligent if we could have more time. For example, we could upgrade this software so that it could give a warning to users when an international student with F1 visa takes courses less than 9 credits in one semester. Or it could report that a student failed in comp exam when s/he try to take the exam again after two times. All would be implemented in Version 2 if we could.

Another possible improvement is to make our GSO management system Internet ready, so that it could provide service to the users within the campus Intranet. To do this, we need to rewrite the code, integrate it with a web server, XML parser and Java Servlet to dynamically generate XML/HTML files.

5. Acknowledgement

Special thanks to Dr. Kalpakis, who instructed us during this semester, gave us a lot of help and encouragement.

Also thank LIST lab, we leveraged two NT machines for development.

Appendix

A.1 List of Source Codes

Index of /software/gso

Name	Last modified	Size	Description
Parent Directory	01-Dec-1999 14:17	-	
AddAttCourses.java	14-Nov-1999 04:14	7k	
AddCompExam.java	26-Nov-1999 02:00	7k	
AddMSProject.java	20-Nov-1999 12:19	8k	
AddNewAdmission.java	28-Nov-1999 15:41	10k	
AddNewApplicant.java	10-Nov-1999 14:17	25k	
AddNewApplicant2.java	10-Nov-1999 18:29	22k	
AddNewApplicant3.java	11-Nov-1999 00:00	17k	
AddNewApplicant4.java	12-Nov-1999 00:31	30k	
AddNewAssistant.java	27-Nov-1999 16:04	11k	
AddNewRA.java	28-Nov-1999 13:07	6k	
AddNewReview.java	26-Nov-1999 13:58	9k	
AddNewStudent.java	13-Nov-1999 23:54	9k	
AddNewTA.java	28-Nov-1999 10:44	9k	
AddPhdPrelim.java	28-Nov-1999 15:19	8k	
AddThesis.java	20-Nov-1999 12:42	8k	
AddTransCourses.java	14-Nov-1999 04:31	8k	
AppDecision.java	28-Nov-1999 14:43	7k	
AppStatus.java	10-Nov-1999 15:18	9k	
Awards.java	11-Nov-1999 22:07	8k	
CMSC661.java	28-Nov-1999 13:41	3k	
CheckUser.java	23-Oct-1999 13:08	1k	
CommitteeMember.java	26-Nov-1999 15:28	5k	
CommitteeMember2.java	25-Nov-1999 22:20	5k	
Default1.prf	28-Nov-1999 16:53	4k	
Dismission.java	12-Nov-1999 14:39	6k	
FacultyFrame.java	06-Dec-1999 17:18	10k	
FrameAbout.java	06-Dec-1999 17:15	2k	
FrameEnterGSO.java	04-Dec-1999 13:40	7k	
FrameSsnWarning.java	09-Nov-1999 13:12	1k	
FrameSuccess.java	09-Nov-1999 13:11	2k	
FrameWarning.java	09-Nov-1999 13:10	2k	
GSO.jpr	30-Nov-1999 22:17	7k	
Graduation.java	11-Nov-1999 15:07	7k	
MainFrame.java	21-Oct-1999 15:37	1k	
ManageCmscCourses.java	14-Nov-1999 03:40	5k	
ManageCourses.java	11-Nov-1999 10:20	4k	
ManageDenyReason.java	09-Nov-1999 11:00	6k	
ManageInstit.java	09-Nov-1999 11:02	8k	
ManagePrerequest.java	09-Nov-1999 11:02	6k	
ManageResKey.java	09-Nov-1999 11:02	6k	
ManageUser.java	04-Dec-1999 12:48	10k	
Manual.java	06-Dec-1999 16:56	2k	
ModifyAdmission.java	25-Nov-1999 23:59	4k	
ModifyAdmission2.java	26-Nov-1999 00:27	12k	
ModifyAppBasicInfo.java	10-Nov-1999 12:39	4k	
ModifyAppBasicInfo2...>	13-Nov-1999 18:23	28k	

```
ModifyAppEduInfo.java   10-Nov-1999 10:34     4k
ModifyAppEduInfo2.java  13-Nov-1999 21:33    25k
ModifyAppOtherInfo.java 11-Nov-1999 00:16     4k
ModifyAppOtherInfo2...> 04-Dec-1999 15:00     8k
ModifyAppScoreInfo.java 10-Nov-1999 22:54     4k
ModifyAppScoreInfo2...> 13-Nov-1999 21:33    19k
ModifyAssistant.java    27-Nov-1999 22:23     4k
ModifyAssistant2.java   28-Nov-1999 15:02    17k
ModifyAttCourses.java   14-Nov-1999 02:40     4k
ModifyAttCourses2.java  20-Nov-1999 10:54     9k
ModifyCompExam.java     26-Nov-1999 01:28     4k
ModifyCompExam2.java    26-Nov-1999 02:19     9k
ModifyMSProject.java    20-Nov-1999 09:39     4k
ModifyMSProject2.java   20-Nov-1999 21:22    12k
ModifyPhdPrelim.java    20-Nov-1999 12:28     4k
ModifyPhdPrelim2.java   26-Nov-1999 02:01    12k
ModifyRA.java           28-Nov-1999 11:58     4k
ModifyRA2.java          28-Nov-1999 15:12     9k
ModifyReview.java       26-Nov-1999 14:38     4k
ModifyReview2.java      26-Nov-1999 14:45     9k
ModifyStudent.java      14-Nov-1999 00:12     4k
ModifyStudent2.java     14-Nov-1999 01:20    10k
ModifyTA.java           28-Nov-1999 10:46     4k
ModifyTA2.java          28-Nov-1999 12:19    12k
ModifyThesis.java       20-Nov-1999 11:40     4k
ModifyThesis2.java      20-Nov-1999 21:24    13k
ModifyTransCourses.java 14-Nov-1999 04:38     4k
ModifyTransCourses2...> 20-Nov-1999 10:53    10k
PersonDataFrame.java    05-Nov-1999 22:10    12k
Publication.java        11-Nov-1999 21:44     8k
SSNInput.java           10-Nov-1999 12:39     4k
StaffFrame.java         06-Dec-1999 17:06    36k
StandardMenu.java       21-Oct-1999 15:49     3k
StudentAwardAdd.java    04-Dec-1999 13:38     8k
StudentAwardView.java   04-Dec-1999 14:37     6k
StudentCompAdd.java     04-Dec-1999 13:13     7k
StudentCompView.java    04-Dec-1999 13:52     6k
StudentFrame.java       06-Dec-1999 17:20     8k
StudentMSAdd.java       04-Dec-1999 13:30     8k
StudentMSView.java      04-Dec-1999 14:24    11k
StudentPhdAdd.java      04-Dec-1999 13:26     8k
StudentPhdView.java     04-Dec-1999 14:00    11k
StudentPublicationAd..> 04-Dec-1999 13:45     8k
StudentPublicationVi..> 04-Dec-1999 15:03     8k
StudentThesisAdd.java   04-Dec-1999 13:33     8k
StudentThesisView.java  04-Dec-1999 14:32    12k
SupperStaff.java        11-Nov-1999 11:39     5k
TADismiss.java          28-Nov-1999 13:10     4k
UserCheck.java          19-Oct-1999 19:43     1k
ViewAdmission.java      26-Nov-1999 00:02     4k
ViewAdmission2.java     26-Nov-1999 00:28    10k
ViewAppBasicInfo.java   12-Nov-1999 10:08     4k
ViewAppBasicInfo2.java  13-Nov-1999 18:34    31k
ViewAppDecision.java    11-Nov-1999 16:27     4k
ViewAppDecision2.java   28-Nov-1999 14:45     5k
ViewAppEduInfo.java     12-Nov-1999 10:23     4k
```

ViewAppEduInfo2.java 12-Nov-1999 10:33 25k
ViewAppOtherInfo.java 12-Nov-1999 11:47 4k
ViewAppOtherInfo2.java 13-Nov-1999 17:39 6k
ViewAppScoreInfo.java 12-Nov-1999 11:33 4k
ViewAppScoreInfo2.java 12-Nov-1999 11:39 18k
ViewAppStatus.java 10-Nov-1999 18:34 4k
ViewAppStatus2.java 10-Nov-1999 18:36 5k
ViewAssistant.java 28-Nov-1999 00:14 4k
ViewAssistant2.java 28-Nov-1999 00:22 12k
ViewAttCourses.java 14-Nov-1999 04:03 4k
ViewAttCourses2.java 14-Nov-1999 04:12 9k
ViewAward.java 12-Nov-1999 01:22 4k
ViewAward2.java 13-Nov-1999 18:40 5k
ViewCompExam.java 26-Nov-1999 02:18 4k
ViewCompExam2.java 26-Nov-1999 02:29 6k
ViewDismission.java 12-Nov-1999 14:37 4k
ViewDismission2.java 12-Nov-1999 19:05 3k
ViewGraduation.java 12-Nov-1999 11:55 4k
ViewGraduation2.java 12-Nov-1999 19:04 6k
ViewMSProject.java 20-Nov-1999 11:05 4k
ViewMSProject2.java 20-Nov-1999 20:31 11k
ViewPhdPrelim.java 20-Nov-1999 12:34 4k
ViewPhdPrelim2.java 20-Nov-1999 20:32 11k
ViewPublication.java 13-Nov-1999 21:55 4k
ViewPublication2.java 13-Nov-1999 21:30 8k
ViewRA.java 28-Nov-1999 12:21 4k
ViewRA2.java 28-Nov-1999 12:34 6k
ViewReview.java 26-Nov-1999 14:48 4k
ViewReview2.java 26-Nov-1999 15:12 7k
ViewStudent.java 14-Nov-1999 00:01 4k
ViewStudent2.java 14-Nov-1999 01:19 8k
ViewTA.java 28-Nov-1999 11:24 4k
ViewTA2.java 28-Nov-1999 12:19 8k
ViewTADismiss.java 28-Nov-1999 12:55 4k
ViewTADismiss2.java 28-Nov-1999 13:12 3k
ViewThesis.java 20-Nov-1999 11:55 4k
ViewThesis2.java 20-Nov-1999 22:13 11k
ViewTransCourses.java 14-Nov-1999 04:57 4k
ViewTransCourses2.java 14-Nov-1999 05:01 9k

A.2 User Manual

User Manual for
Graduate Student Office (GSO) Management System

Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County

Table of Contents

A1. Installation Guide 30
A1.1 System requirements 30
A1.2 Start the application 30
A1.3 Users 30
A2. Functions 31
A3. GSO Staff 32
A3.1 Maintain Basic Information 33
A3.2 Manage Applicant Information 36
A3.3 Manage Student Information 42
A3.4 Manage Graduate Assistant 46
A3.5 Other management 49
A3.6 Help 51
A4. Faculty 52
A4.1 Review applicant 52
A5. Students 56
A5.1 Input student information 56
A5.2 Query student information 59
A6. Other Information and Acknowledgement 63

CMSC 661, Fall 1999
Instructor: Dr. Kalpakis

Team members:
Honglin Su, hosu@cs.umbc.edu
Shilang Tang, stang2@cs.umbc.edu
Zhongli Ding, zding1@cs.umbc.edu

A1. Installation Guide

A1.1 System requirements

The GSO management system was implemented by pure Java, all you need to run the application are the hardware with Java virtual machine (JDK 1.1 or above), we have tested the platforms of Sun SPARC, SGI Indy, and PC (Windows NT, Win95/98). The distribution of the application also includes Oracle JDBC driver. Before you run the application, make sure that you already set your Java running environment, add this distribution file (gso.jar) to your Java CLASSPATH.

e.g. for UNIX:
setenv CLASSPATH /home/grad3/hosu/lib/gso.jar:…

for Windows
set CLASSPATH=c:\shl\lib\gso.jar

A1.2 Start the application

To start the application, simply type the command: %java GSO/CMSC661, a user log in menu will be popped out.

A1.3 Users

We define 3 types of users to use this system. One is the GSO staff, who has the privilege to maintain this sytem; another one is faculty member, who can review the application and view applicant's information; and the third one is the student, who can only input or

query their own relevant information, such as publishing, academic awards, etc. All three types of users shall the same log-in-menu. After the user log in, the system will pop up another menu based on different type of user.

A2. Functions

The users of this database application are going to be personnel from GSO and faculty, and students have a very limited manipulation capability. Most of the modifications to the information in the database will be done by GSO staff, while faculty will be accessing the database mostly in read-only mode, except for the reviewer's input.

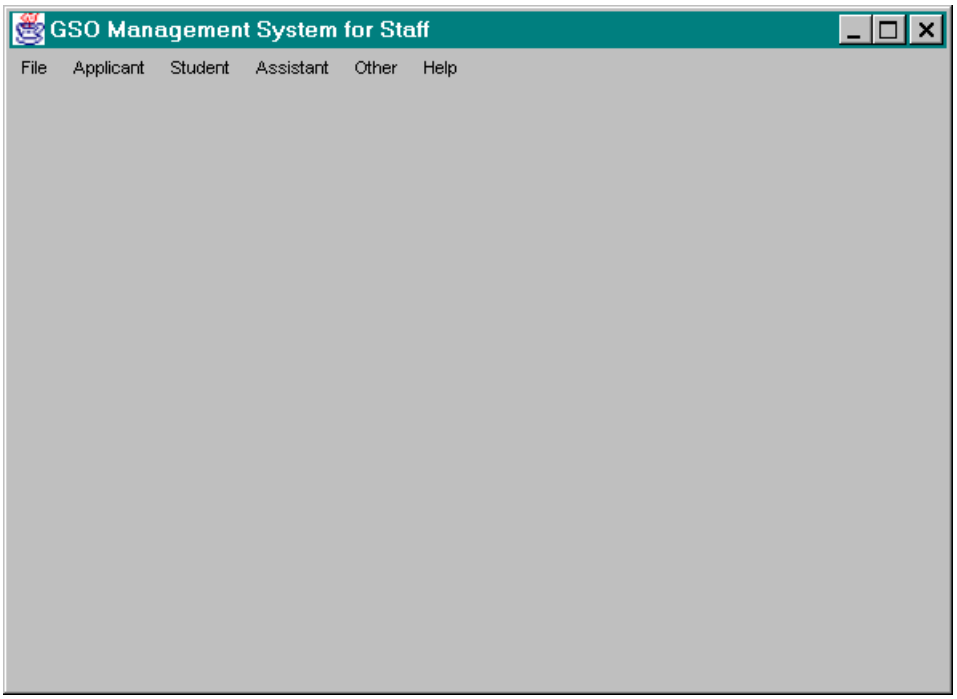
The system handles the following event.

28. Receive a new application, add new personal information into the database;
29. Update personal information when supporting materials are arriving;
30. When all application materials are received, update the status;
31. Application is sent to reviewer, grant appropriate user rights to the reviewer;
32. Reviewer makes a recommendation for the applicant, update the reviewer's table;
33. Application is received from reviewer, revoke the user rights to access the application;
34. Application is sent to admissions committee, grant appropriate user rights to admission committee;
35. Application is received from admissions committee, revoke the user rights to access the application;
36. Committee makes a decision, update the applicant status;
37. Letters are sent (departmental, graduate school, international office), update the dates respectively;
38. Applicant replies, update the applicant information;
39. Admissions committee approves defer request if the defer request is received by the department, update applicant status.
40. Admitted student enrolls in a program, update the student's information. A new row is inserted in the student table.
41. Student change status, update the student information;
42. Student takes classes, update the student and courses relation information;
43. Student changes academic/research advisors, update the student information;
44. Student takes comprehensive exams, add an entry to comprehensive table;

45. Student forms his/her advisory, examination, or defense committees, update the committee information;
46. Student takes Ph.D. preliminary exam, add an entry to the Ph.D. preliminary table;
47. Student defends thesis, update the thesis table;
48. Student completes M.S. project, update the MS. Project table;
49. Student is admitted into Ph.D. candidacy, add an entry to Ph.D. candidacy table;
50. Student receives an award, add an entry to award table;
51. Student is awarded an assistantship, add an entry to assistantship table;
52. Student is dismissed from his/her assistantship, update assistantship table and student status;
53. Student is dismissed from the program, add an entry to dismissed student table;
54. Student graduates, add an entry to graduated student table.

A3. GSO Staff

When GSO staff logs in the system, the following menu is shown. All of the GSO staff operation can be conducted through this menu. It has six main options: File, Applicant, Student, Assistant, Other and Help.



File: Manage the database and exit. From this option, the staff can input the institution information, research keywords, UMBC courses, prerequisite courses, deny reasons, user/faculty management.

Applicant: Manage the applicant related information including receive an application, modify and query applicant information, status, and applicant's reply.

Student: Manage the student related information. After the applicant is accepted, and enrolled at UMBC, he/she will become a student, the functions including modify/query student information, courses' grade, transfer courses, MS project, thesis, Ph.D. preliminary and comprehensive examinations.

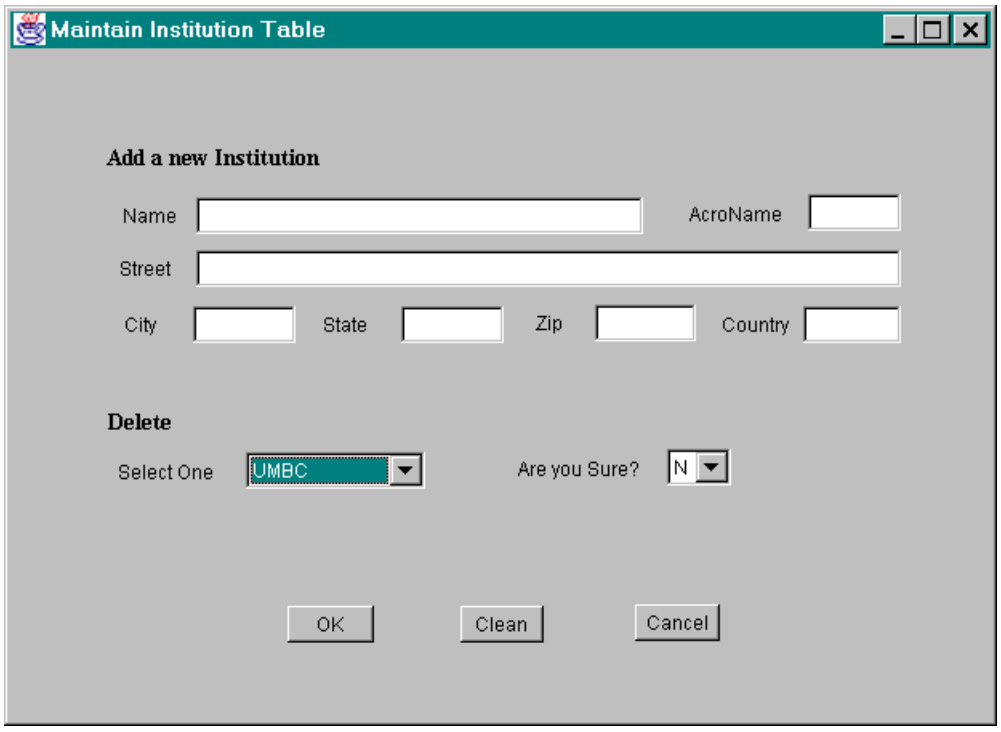
Assistant: Manage the graduate assistant related information.

Other: functions including student graduation, dismission, awards, publications, and semester based course schedule.

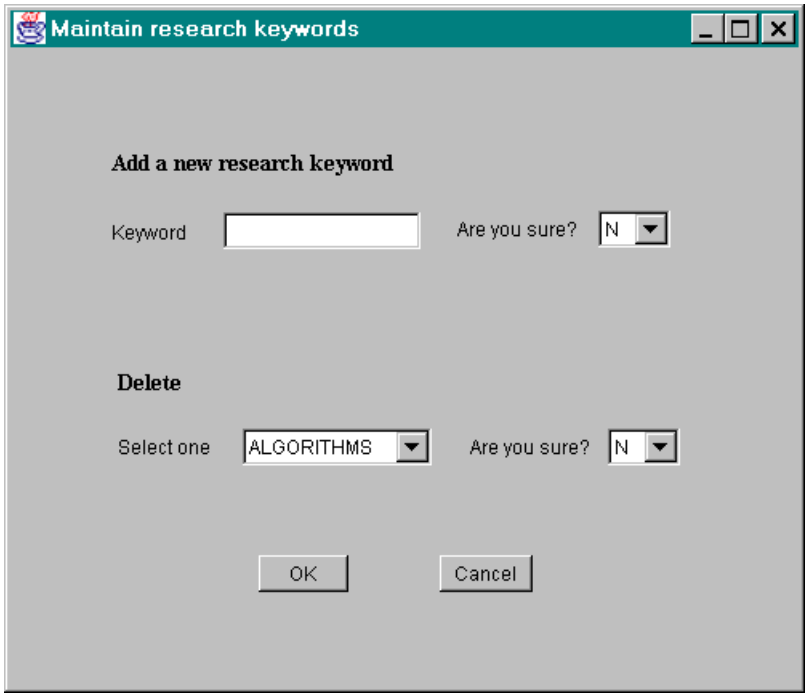
Help: Help information and version and team member of the project.

A3.1 Maintain Basic Information

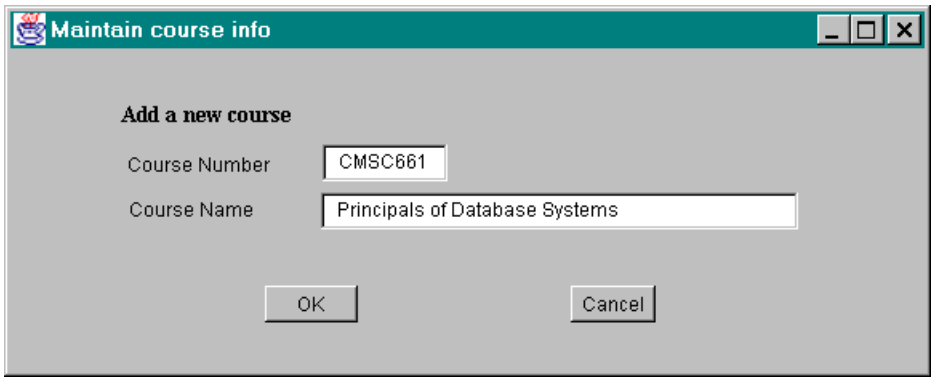
To maintain institution table: staff -> file -> manage DB -> Institution



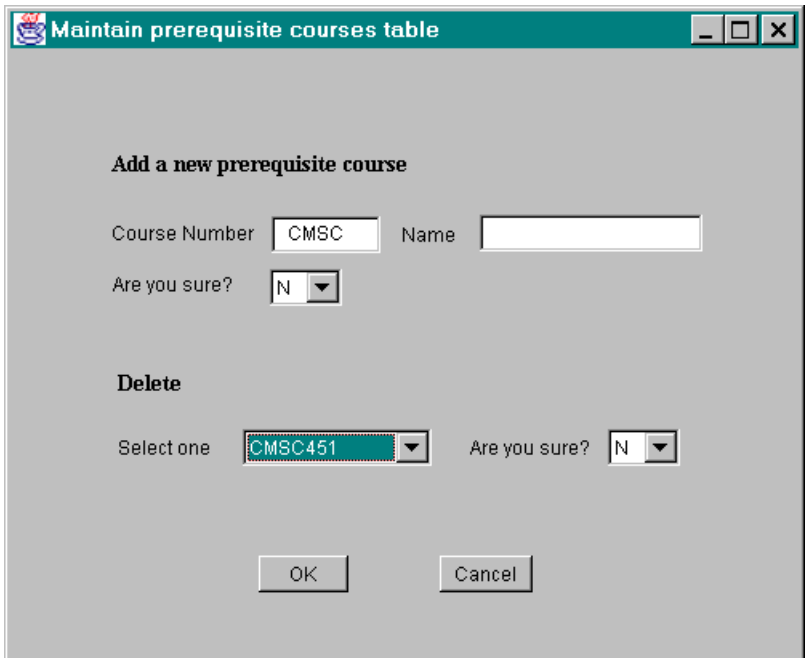
To maintain research keyword, staff -> file -> manage DB -> ResKeyWord



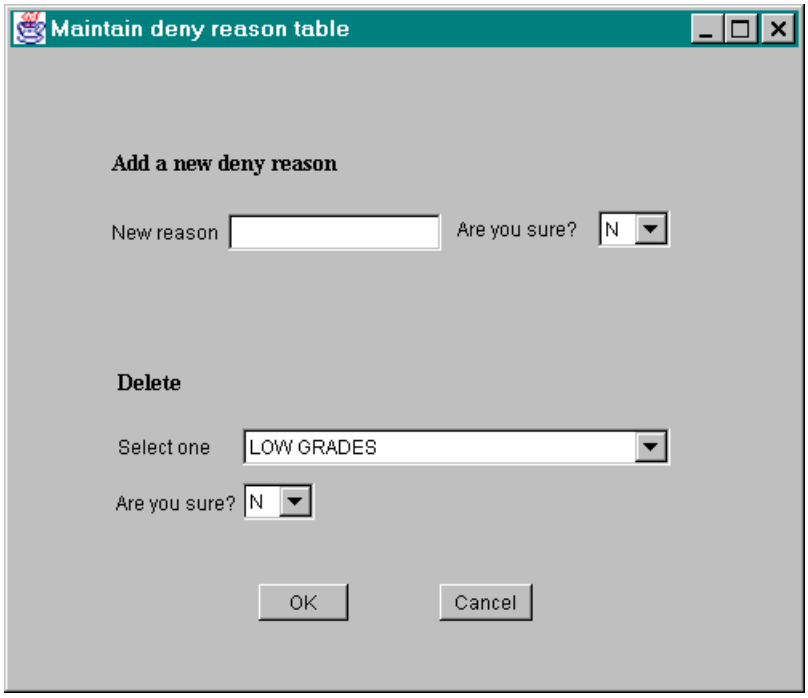
To maintain course information, staff ->file->manage DB->UMBC course



To maintain prerequisite courses information, staff -> file -> manage DB -> prerequisite

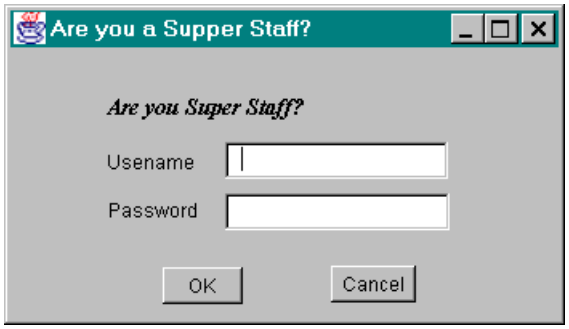


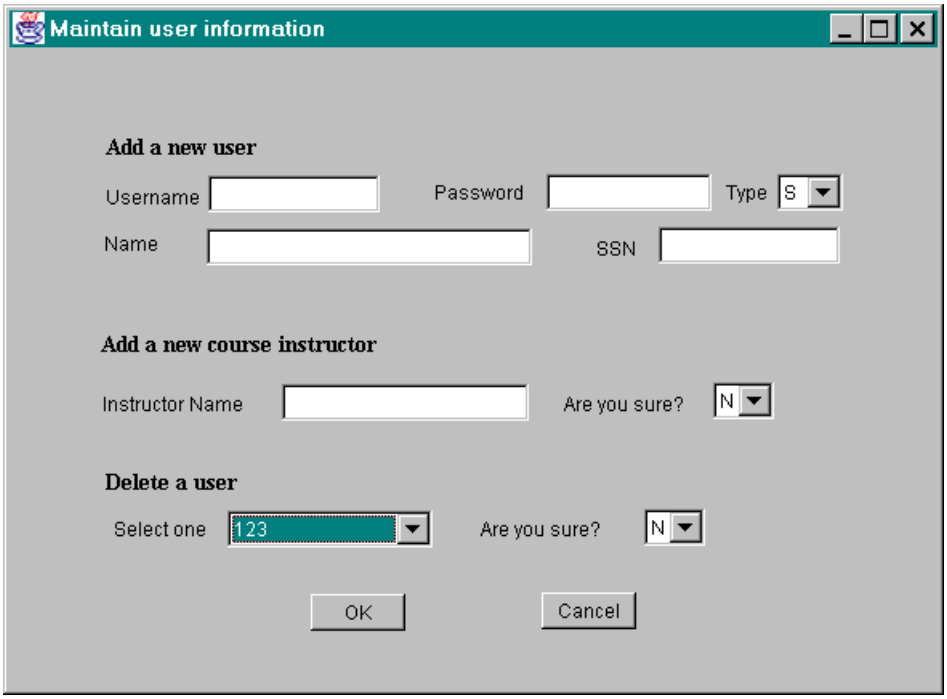
To maintain deny reason table, staff -> file -> Manage DB -> deny reason



To manage user/faculty information, staff -> file -> Manage DB -> user/faculty

only super staff has such privilege to do this, a confirmation menu would be popped up:



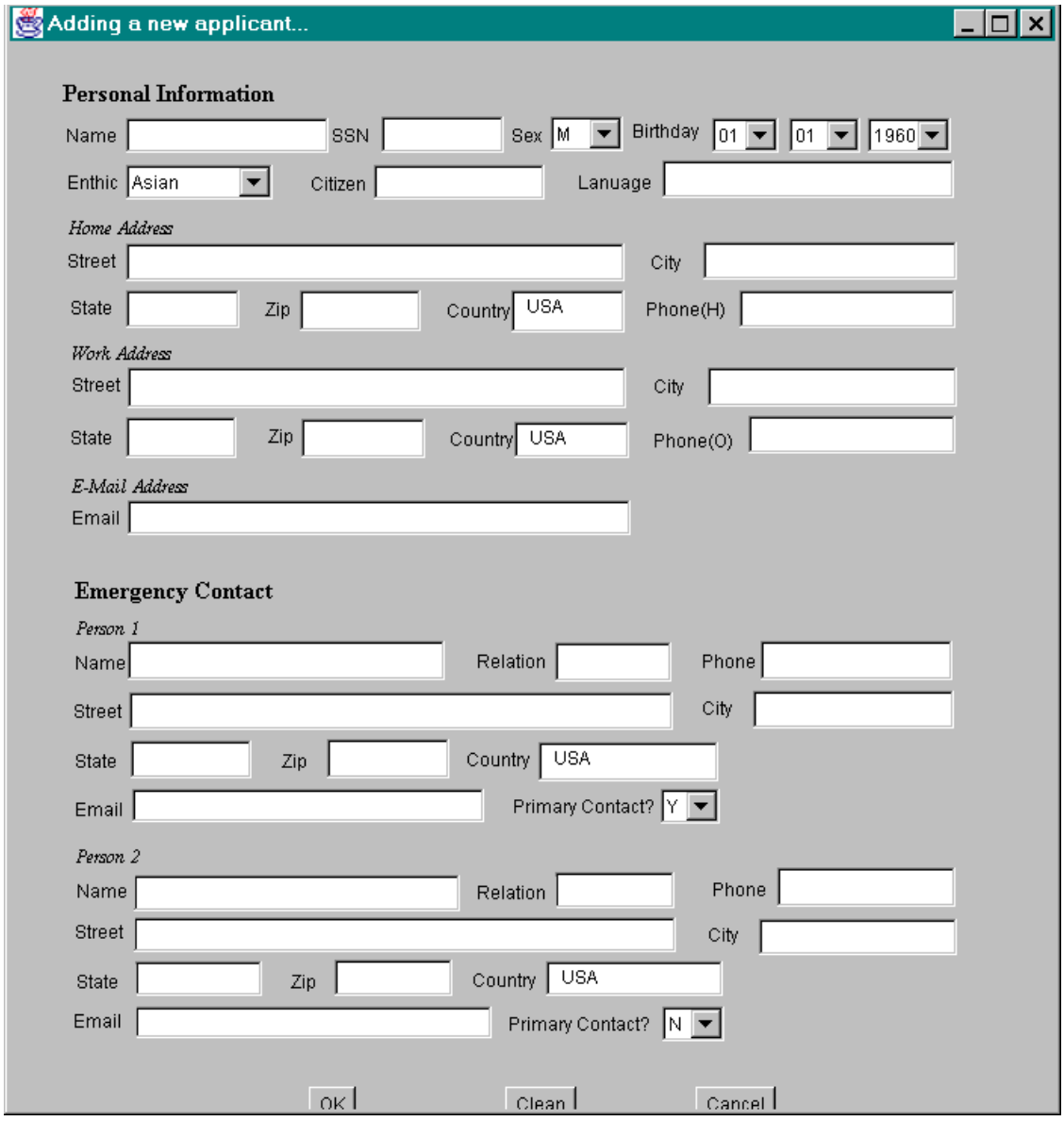


To safely exit the system, staff -> file -> exit.

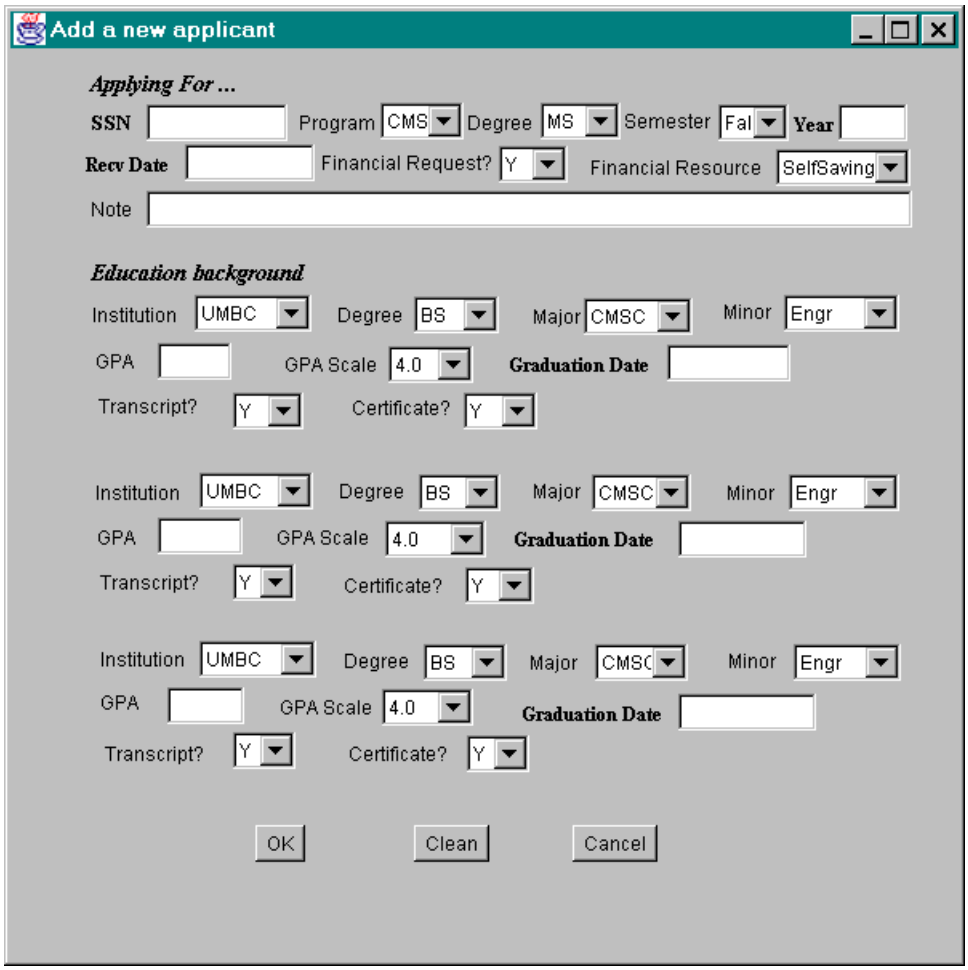
A3.2 Manage Applicant Information

To add a new applicant: staff -> applicant -> receive a document -> Input basic info / Input Education Info / Input Score Info / Input Other Info

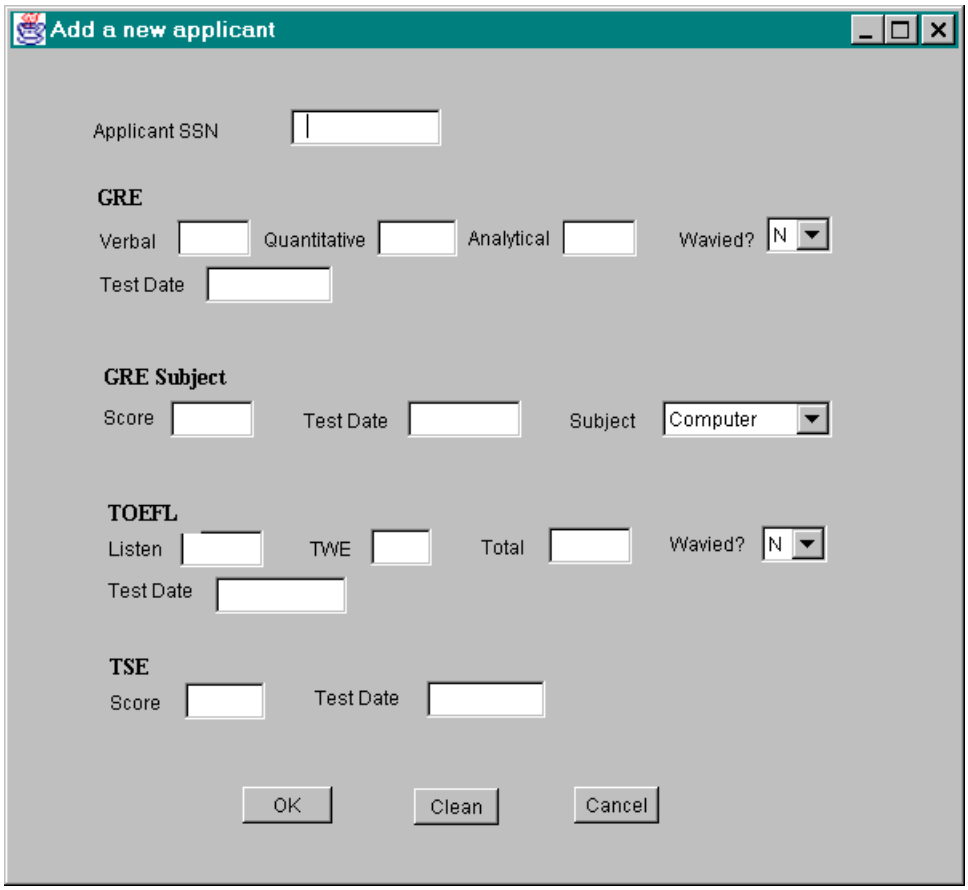
Basic Information:



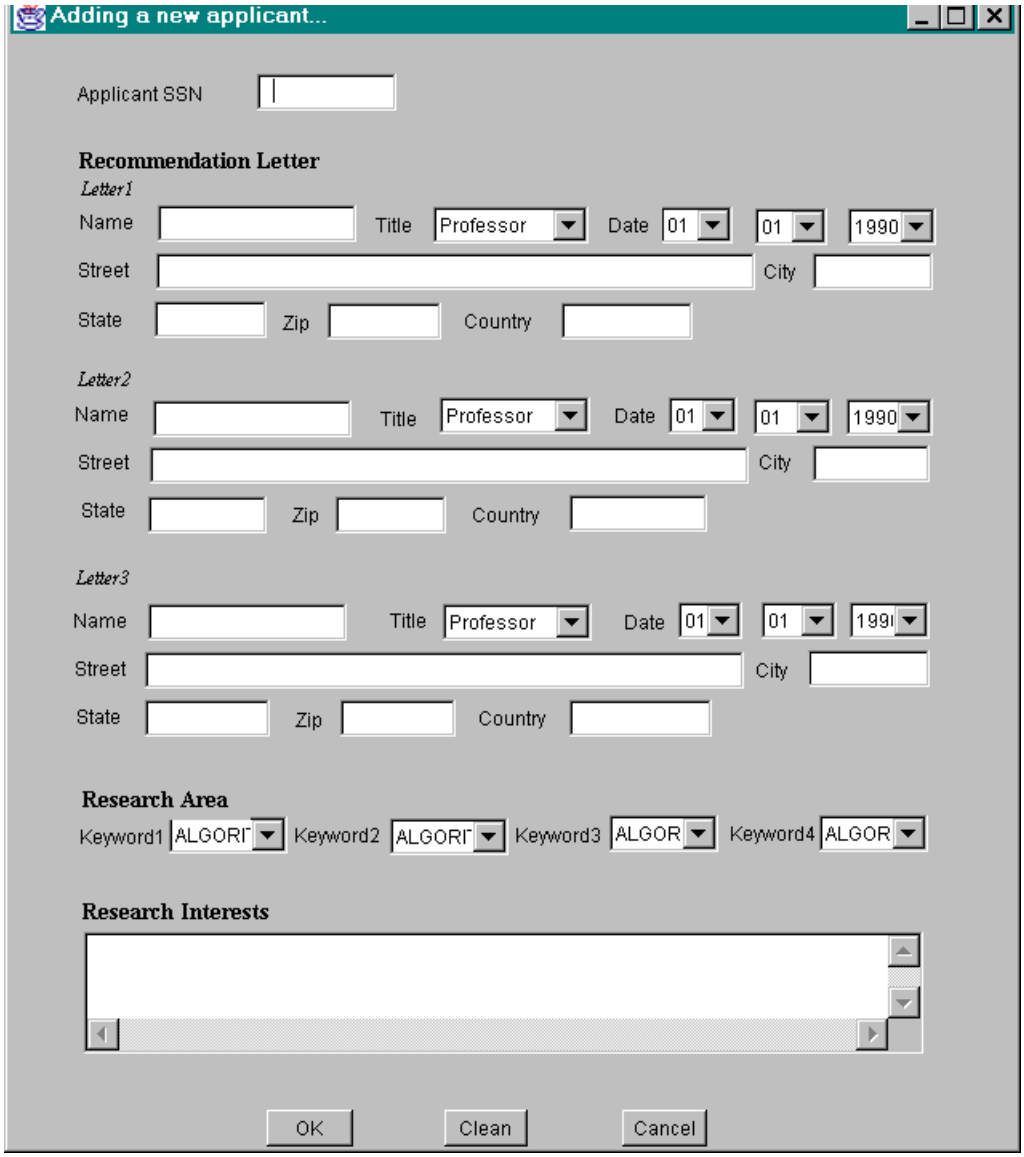
Education Information:



Score Information:

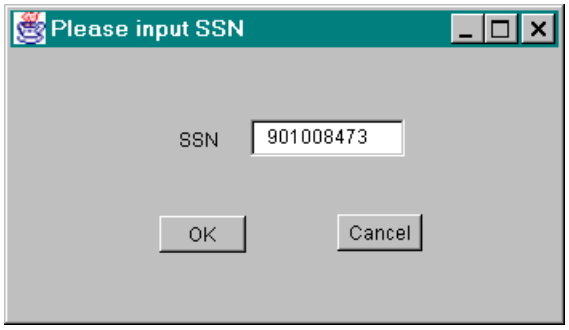


Other Information:



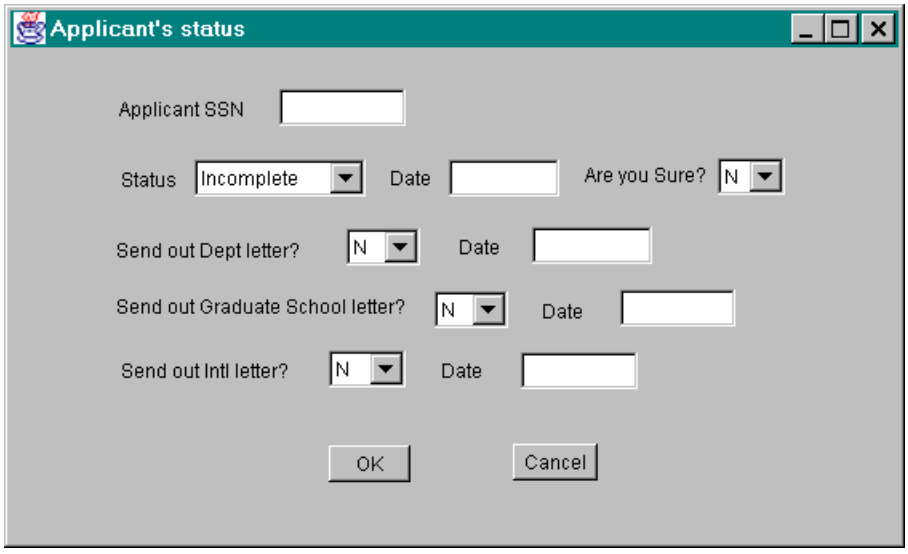
To Modify applicant information, staff -> applicant -> modify -> basic info / Education Info / Score Info / Other Info.

A new menu would be popped, waiting for you to input the applicant SSN.

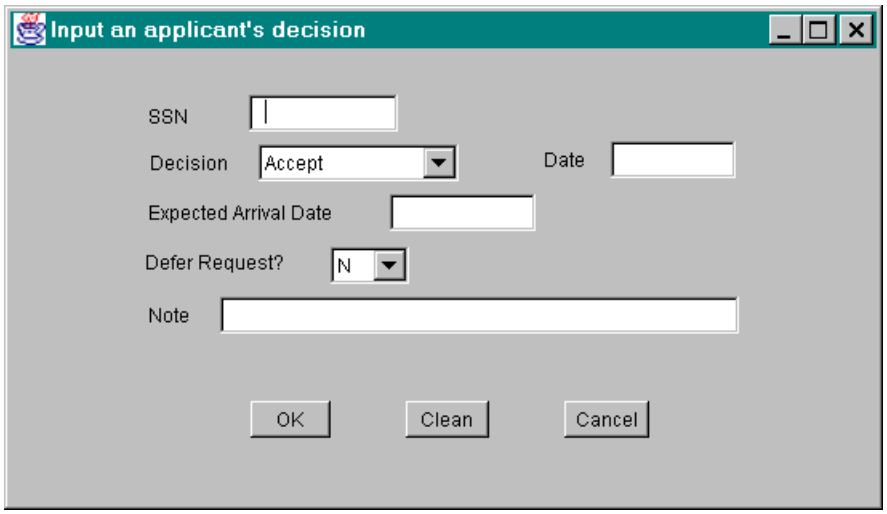


To query applicant information, do it similarly, input the SSN, then view the basic info / Education Info / Score Info / Other Info.

To modify/view applicant's status, staff -> applicant -> applicant status -> modify/view, input the applicant's SSN and do it accordingly.

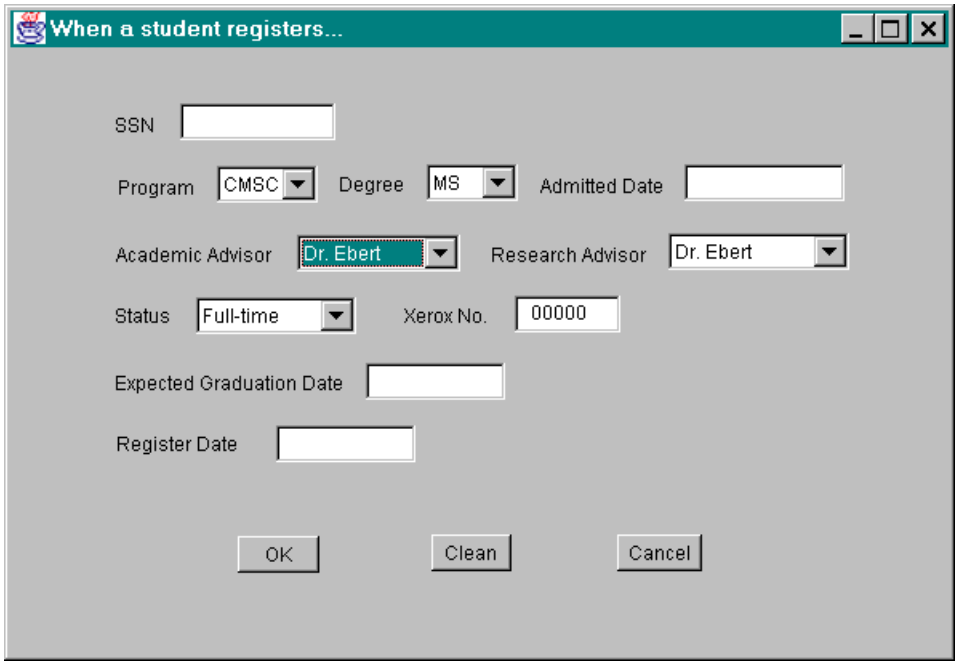


To input/view applicant reply, staff -> applicant -> applicant reply -> input/view

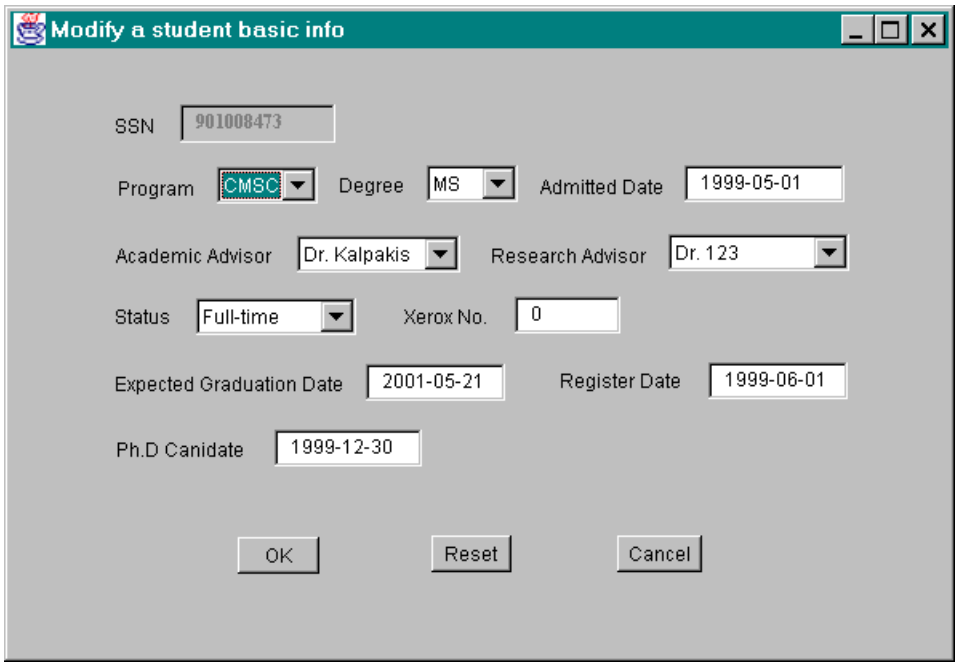


A3.3 Manage Student Information

To add a new student, staff -> student -> add a new student

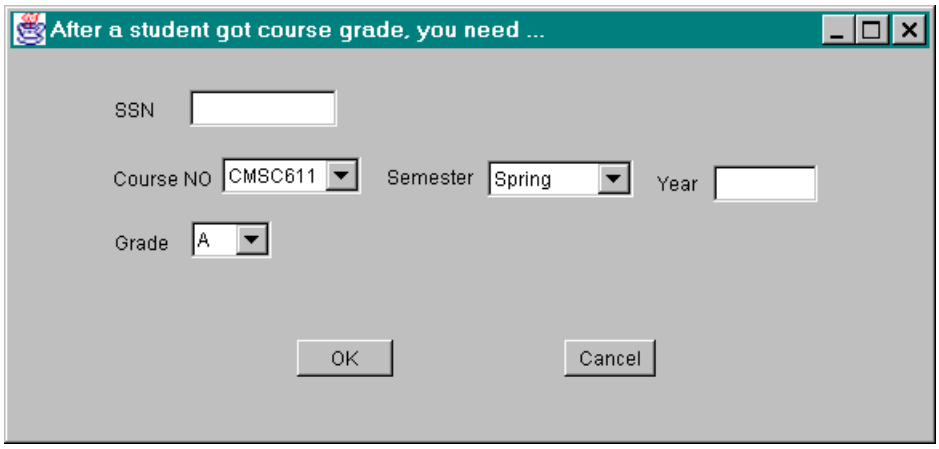


To modify/view student, you need to input SSN, then do it accordingly.

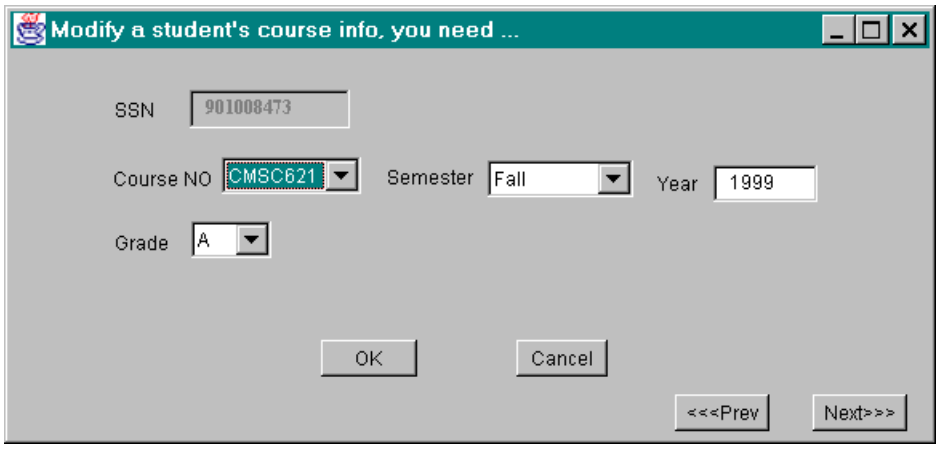


You can add/modify/view student's course grade information, staff -> student -> course grade info -> add/modify/view

Add:

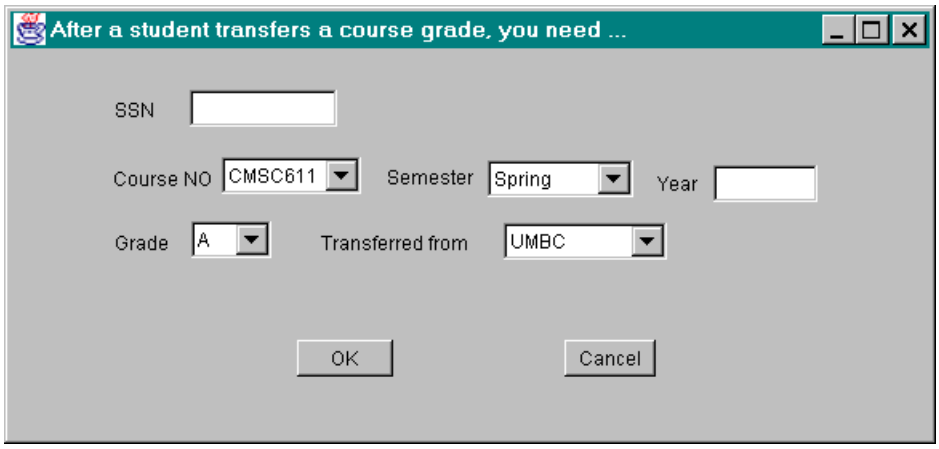


Modify: after input the SSN, you can modify it. You can also use prev and next button for other course grades.

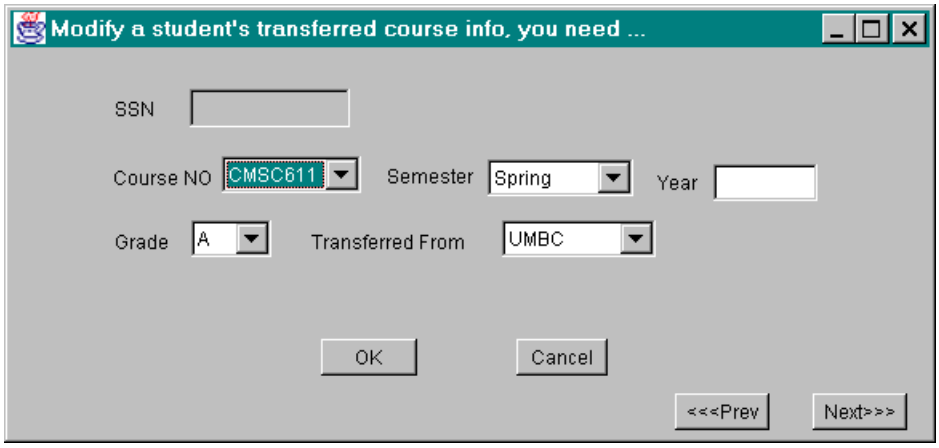


You can also add/modify/view transferred courses for the student. staff -> applicant -> transferred course.

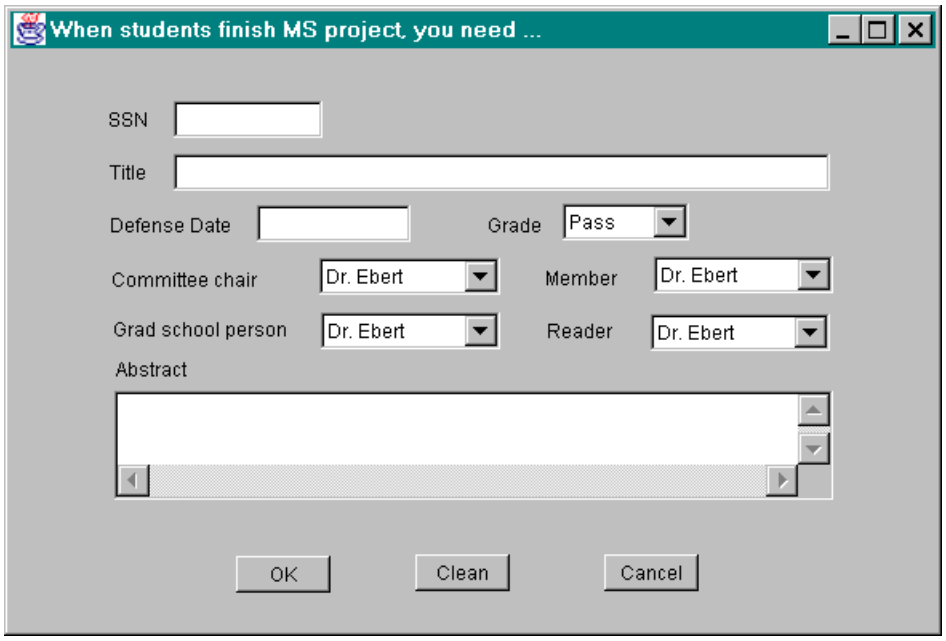
Add:



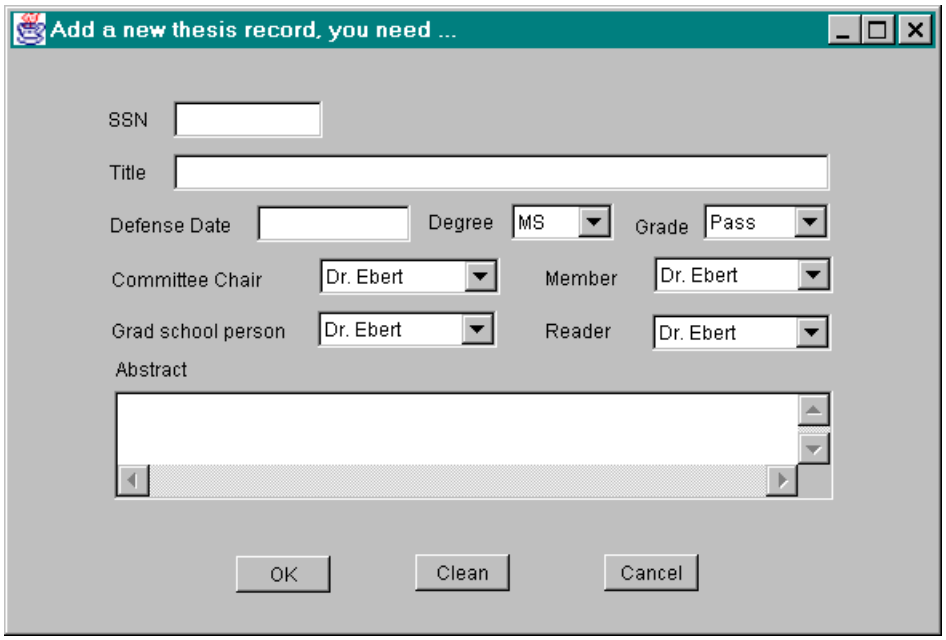
Modify: you need to input the SSN first.



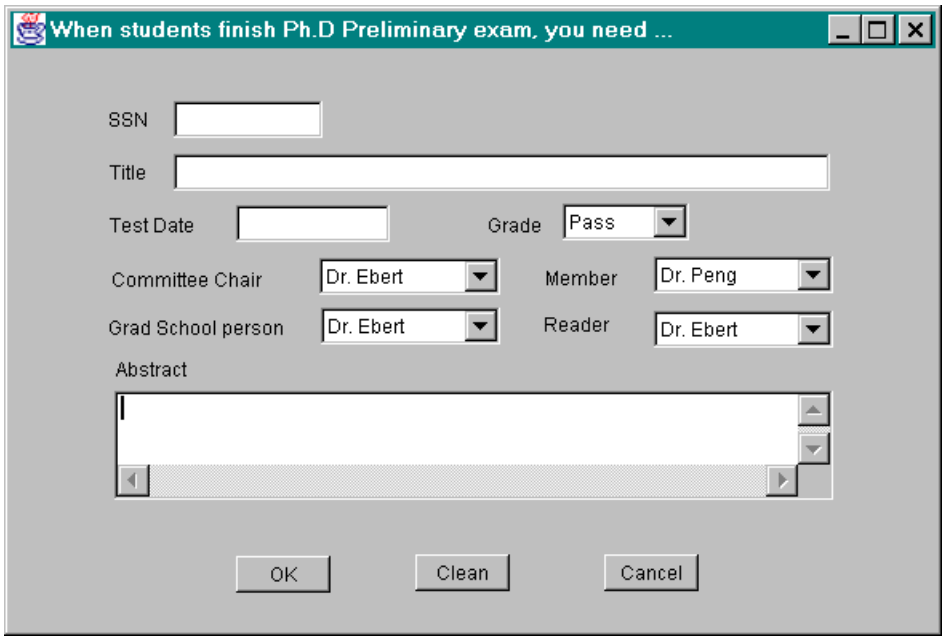
You can add/modify/view student's MS project, staff -> student -> MS project



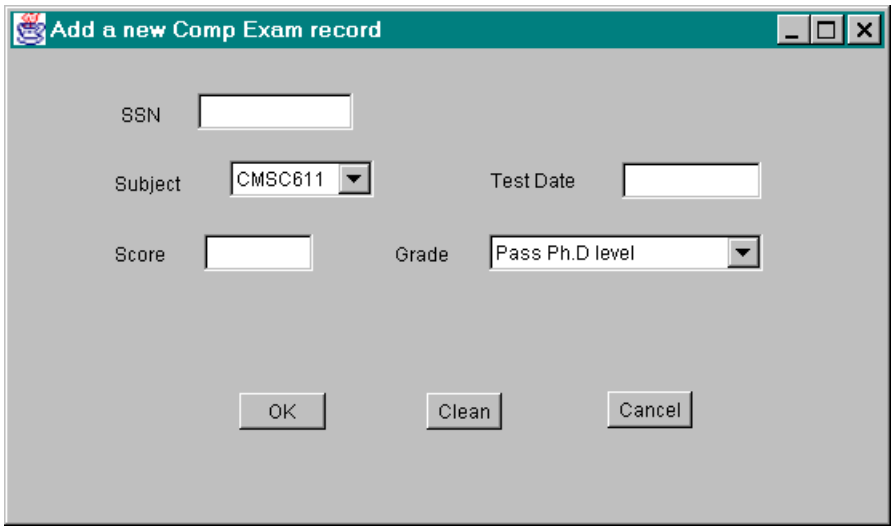
You can add/modify/view student's thesis, staff -> student -> thesis



You can add/modify/view student's Ph.D. preliminary information, staff -> student -> Ph.D. preliminary



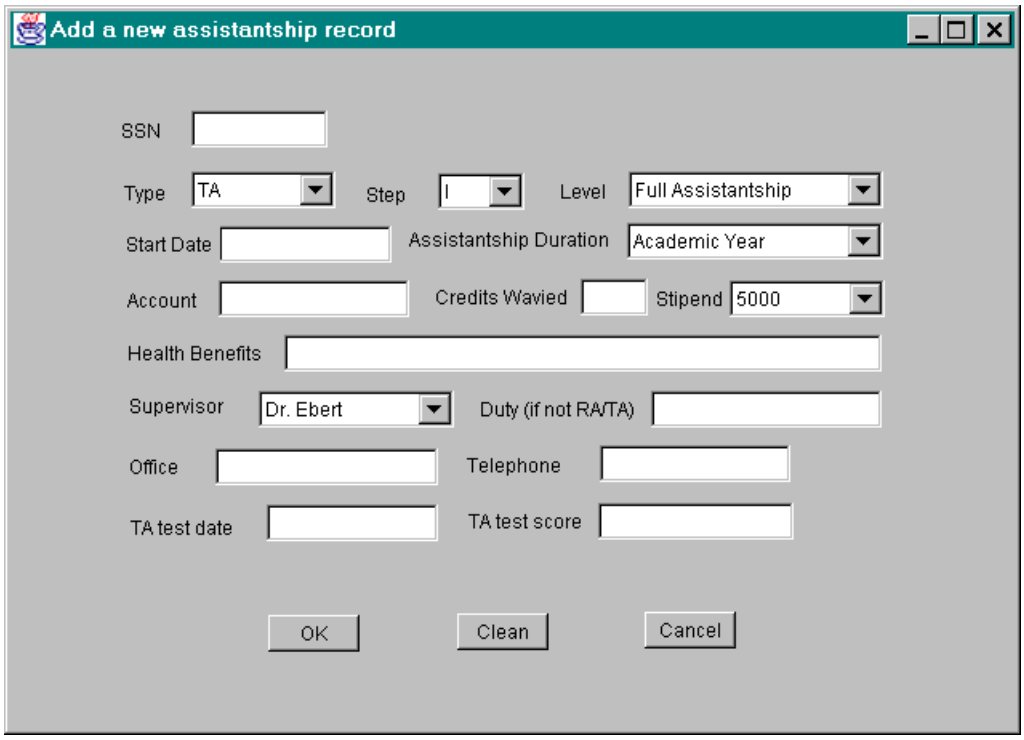
You can add/modify/view student comprehensive exam, staff -> student -> Comp Exam.



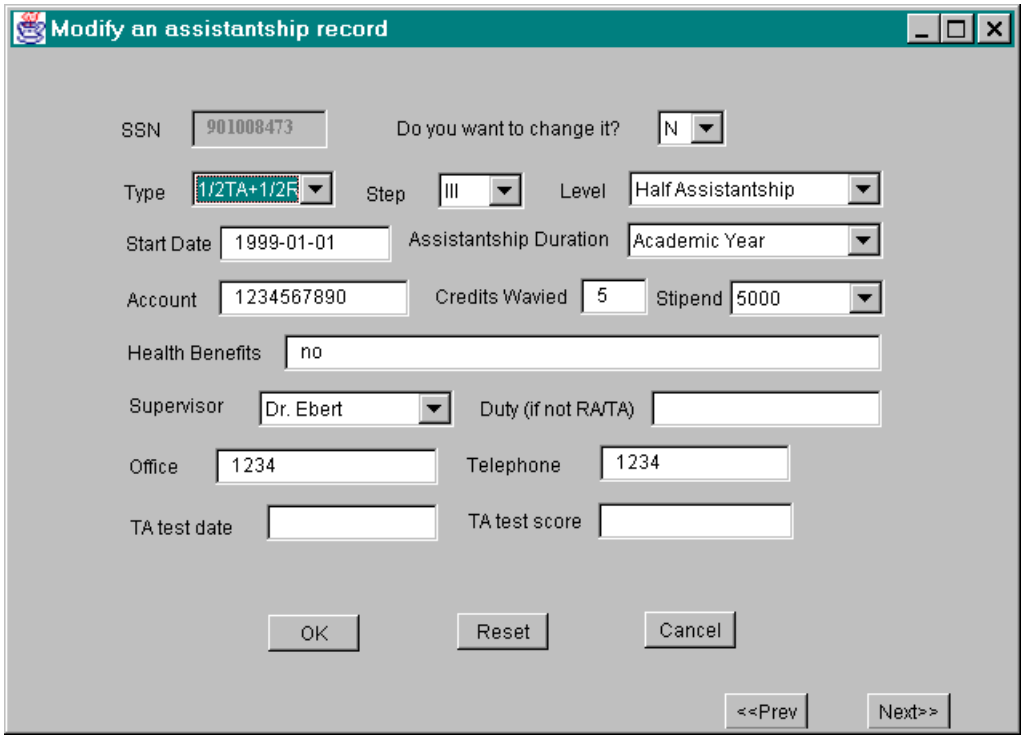
A3.4 Manage Graduate Assistant

You can add/modify/query assistant's information. staff -> assistant -> assistant info.

Add information:

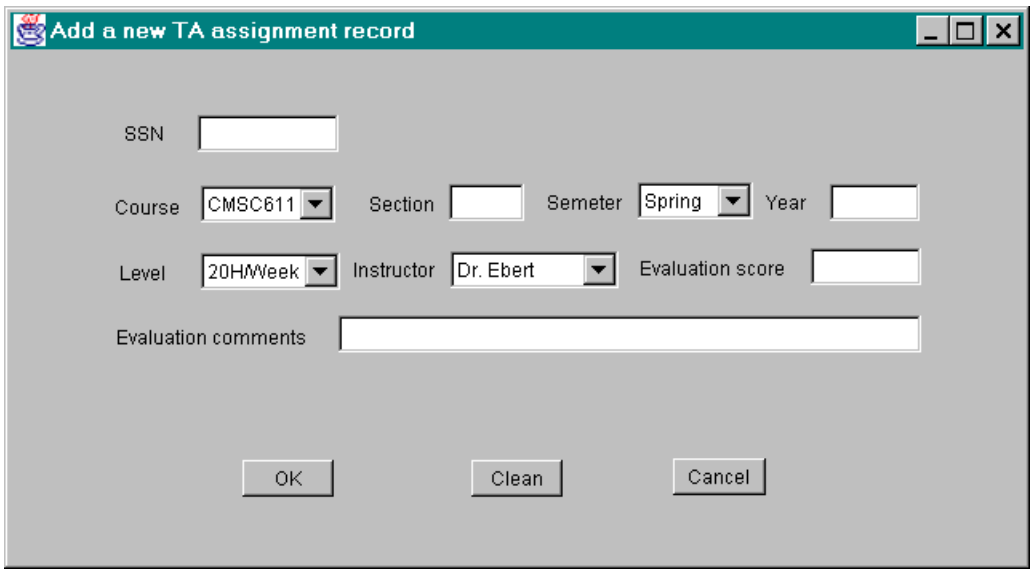


To modify an assistant information, you need to input SSN first, then do it accordingly.

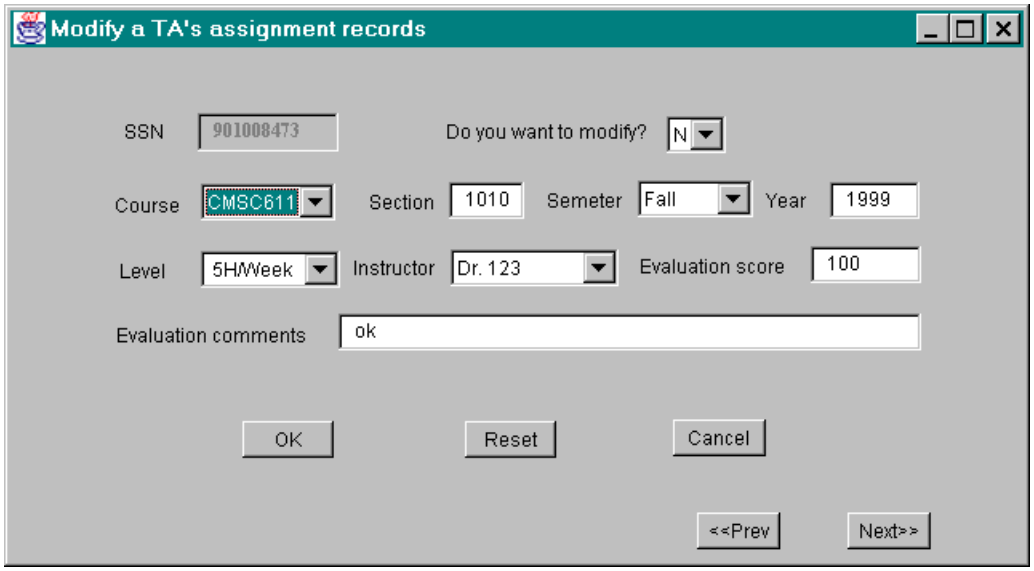


For TA management, staff -> assistant -> TA management

Add a new TA assignment:

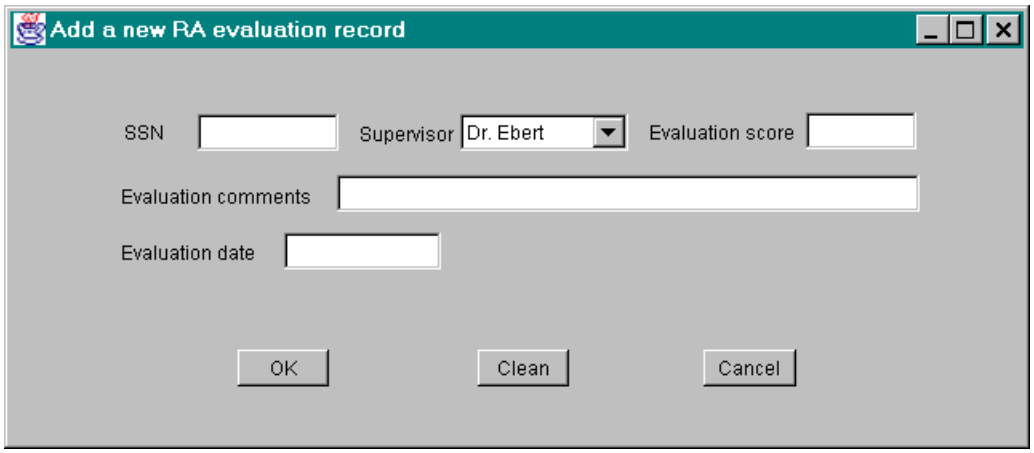


To modify a TA assignment, you need to input the SSN, then do it accordingly.

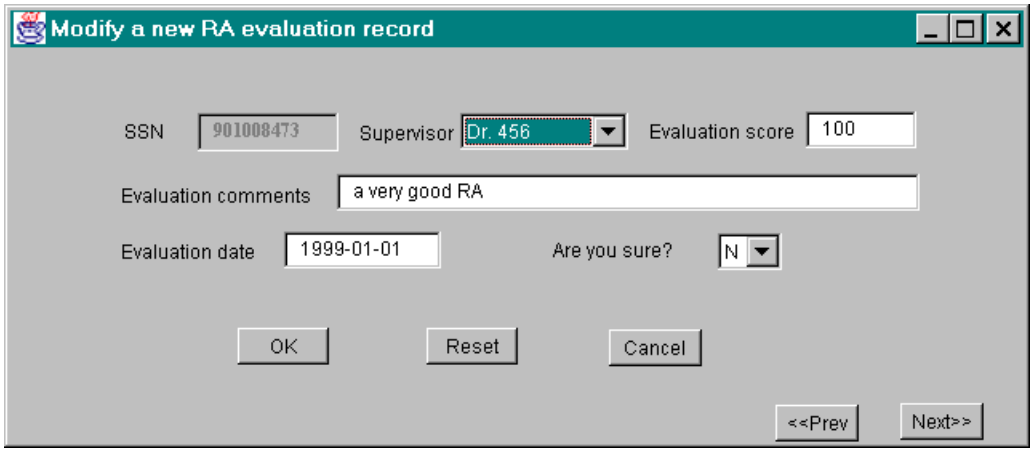


For RA management, you can do evaluation, modify, and query information. staff -> assistant -> RA management

Evaluation:



Modify:

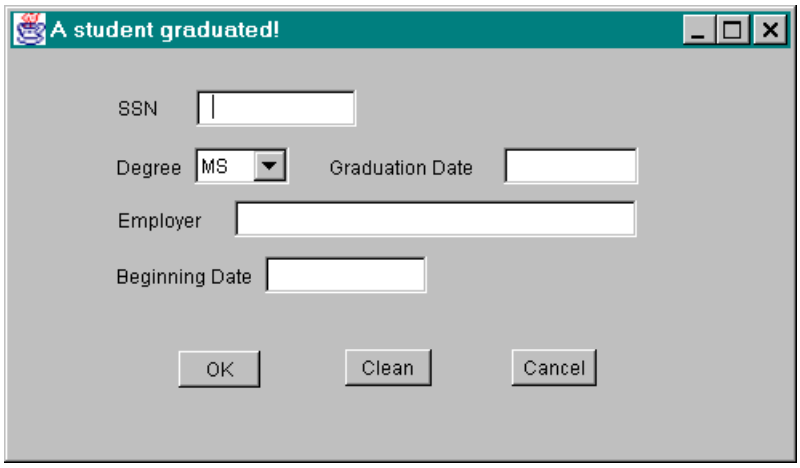


To dismiss an assistant, staff -> assistant -> dismiss

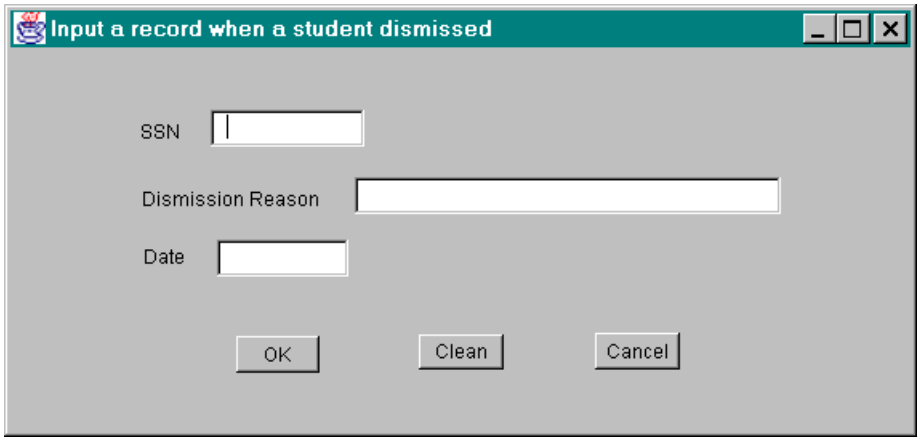
A3.5 Other management

In this part, the system keeps the records of graduated student, dismissed students, student's awards, publications and course schedule.

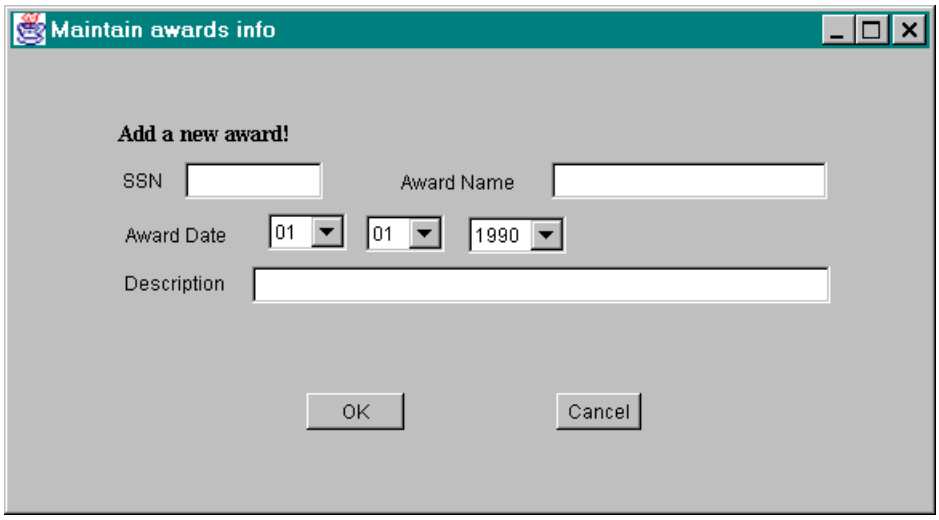
You can add/view students, who are graduated. staff -> other -> Student Graduation.



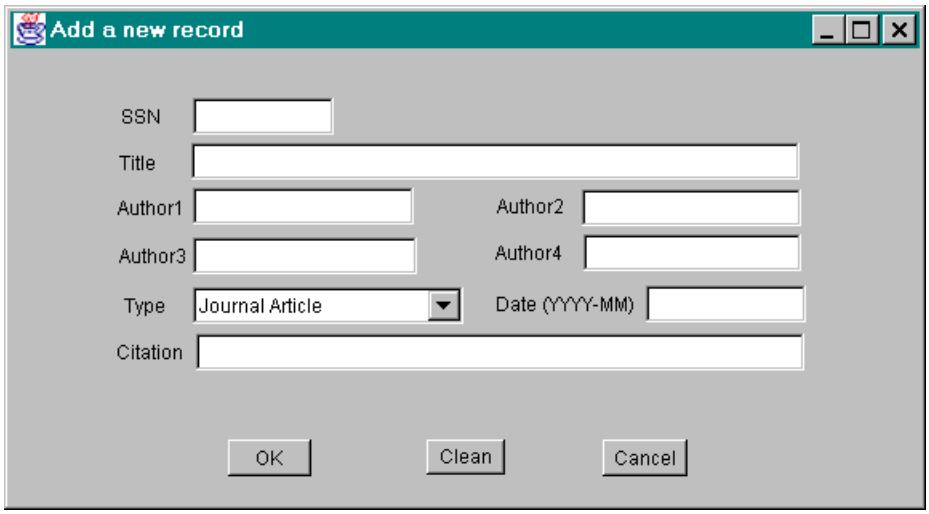
You can add/view students, who are dismissed. staff -> other -> Student Dismissed.



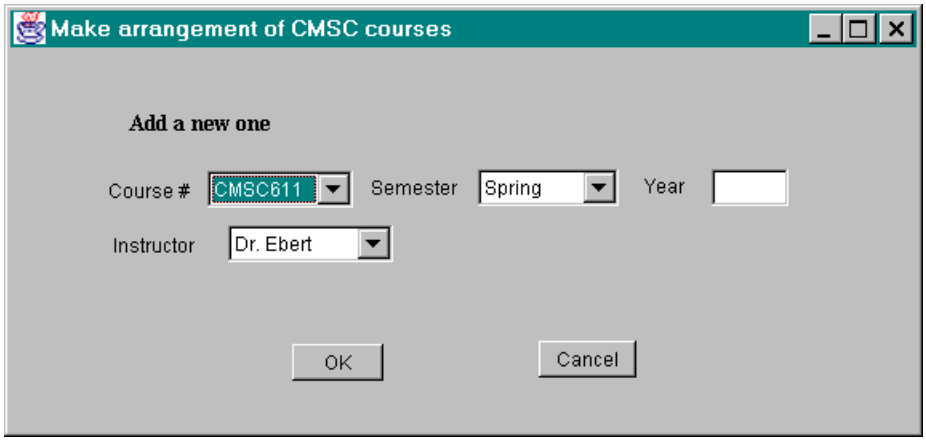
You can add/view student's awards. staff -> other -> Awards



You can add/view student's publications. staff -> other -> Publications



And, the staff can add the course schedule for specific semester. staff -> other -> course schedule.

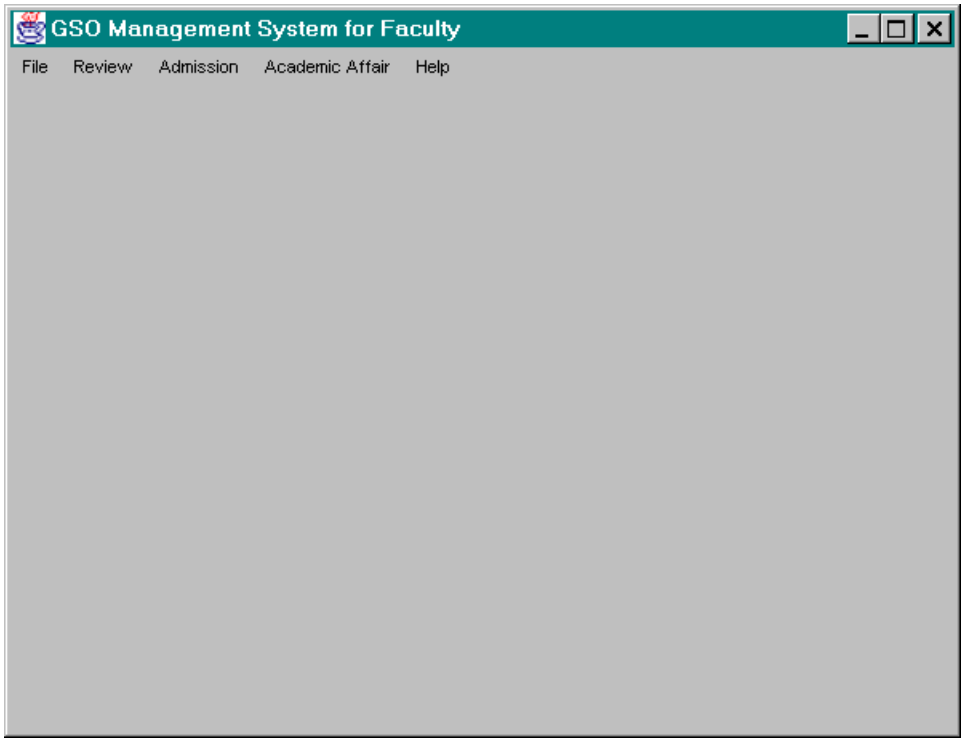


A3.6 Help

The system gives help option for the user, and the version information of the system.

A4. Faculty

Faculty share the same log in menu as GSO staff, however, after faculty log in, a different menu will be shown.



There are five major functions in this part.

File: exit the system safely.

Review: review applicant's information and make decision.

Admission: grant admission to applicant.

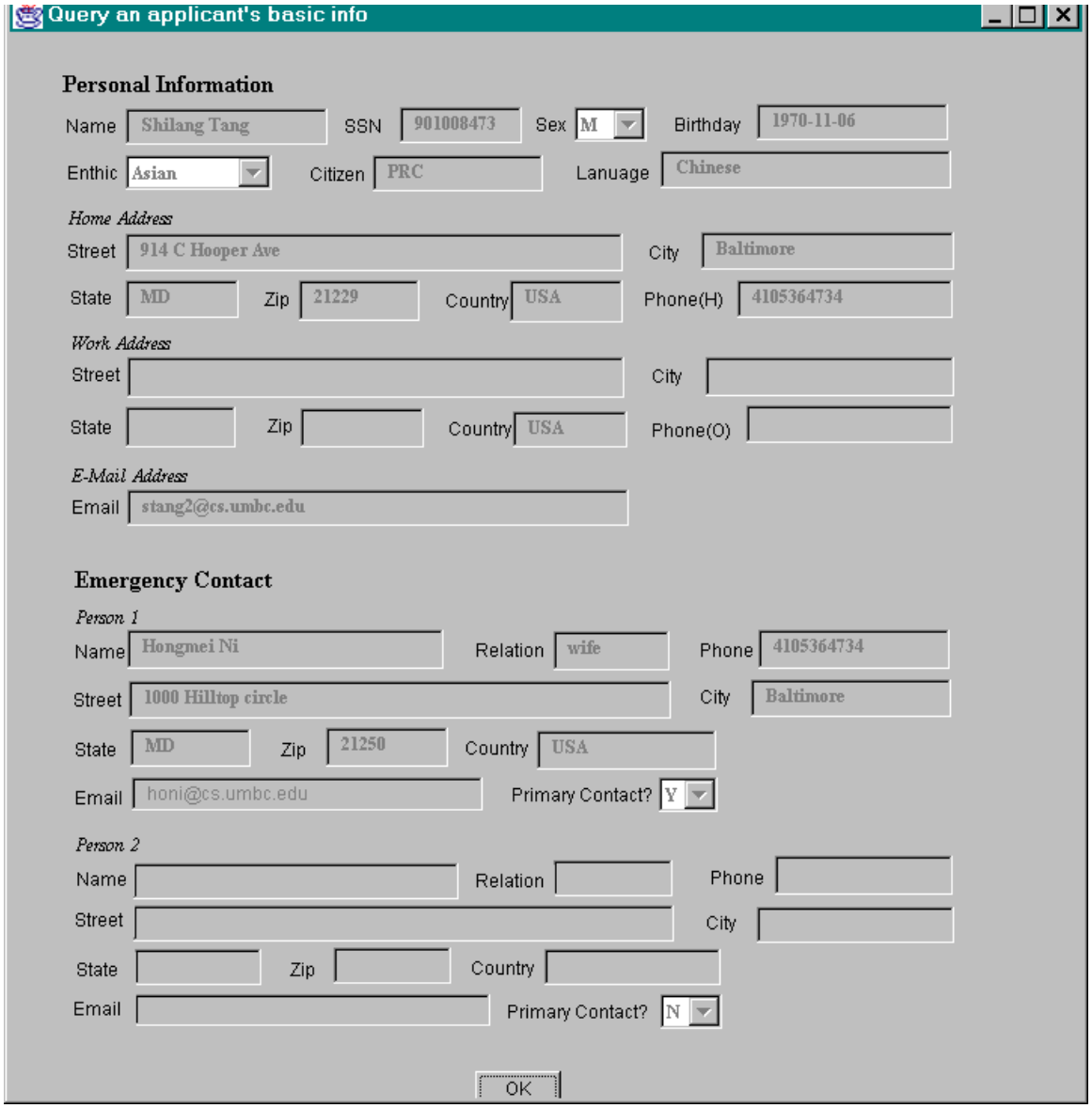
Academic affairs: students' awards, publications, MS project, comp exams, Ph.D. preliminary and thesis.

Help: manual and system versions.

A4.1 Review applicant

Faculty can review applicant basic, education, score and recommendations letters information, based on applicant SSN. However, faculty can only query application information, they can not modify such information.

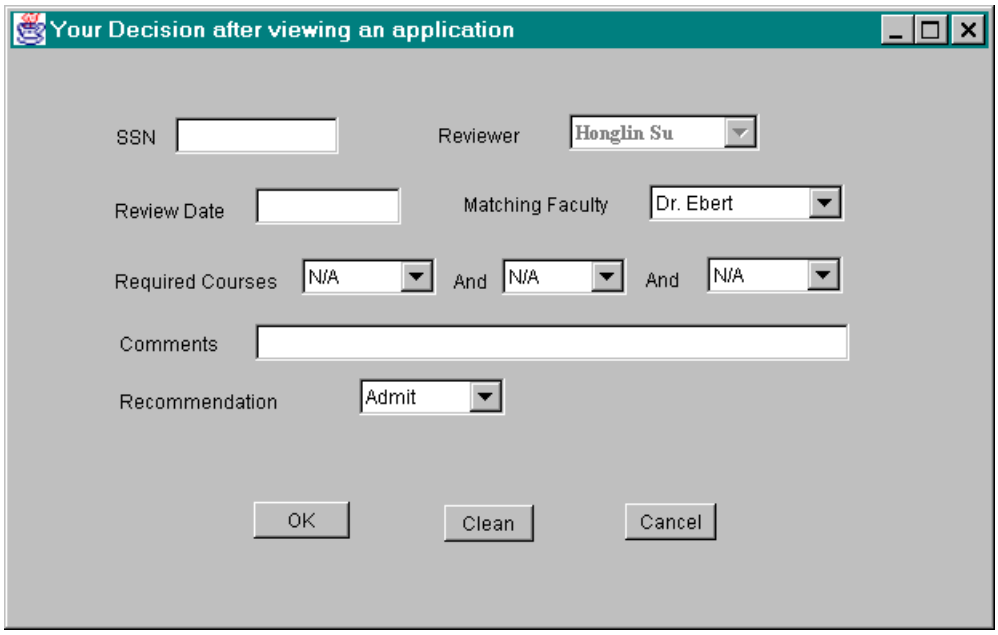
faculty -> review -> Applicant basic info / education info / score info / recommendation letters.


Query an applicant's basic info
Personal Information
Name Shilang Tang SSN 901008473 Sex M Birthday 1970-11-06
Enthic Asian Citizen PRC Lanuage Chinese
Home Address
Street 914 C Hooper Ave City Baltimore
State MD Zip 21229 Country USA Phone(H) 4105364734
Work Address
Street City
State Zip Country USA Phone(O)
E-Mail Address
Email stang2@cs.umbc.edu
Emergency Contact
Person 1
Name Hongmei Ni Relation wife Phone 4105364734
Street 1000 Hilltop circle City Baltimore
State MD Zip 21250 Country USA
Email honi@cs.umbc.edu Primary Contact? Y
Person 2
Name Relation Phone
Street City
State Zip Country
Email Primary Contact? N
OK

Faculty can make/modify/view their own decision to the applicant.

faculty -> review -> your decision -> Make new decision / modify decision / view decision

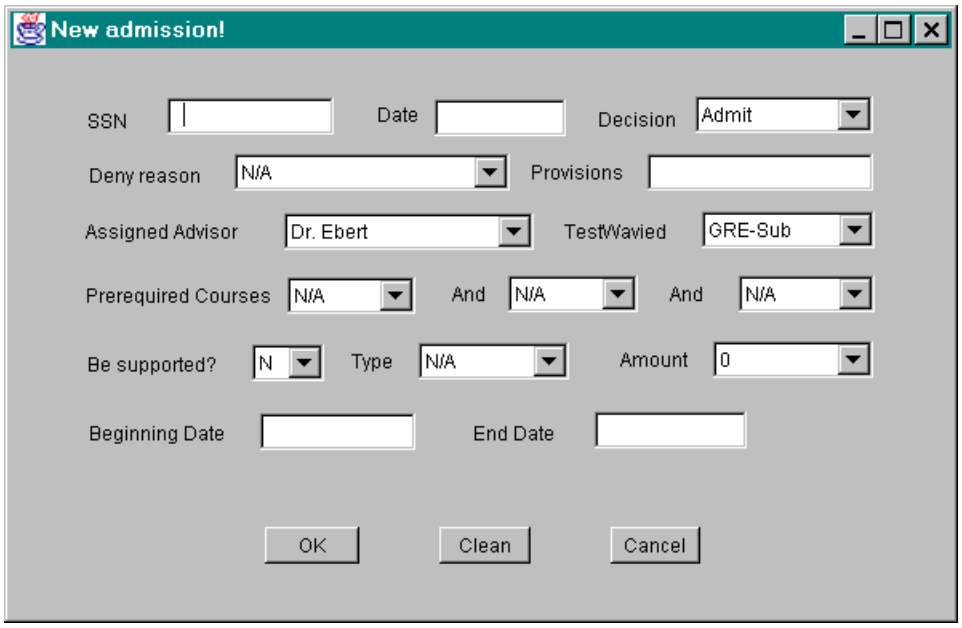
Make new decision:



For faculty, they can grant admission to applicant, if they are admission committee.

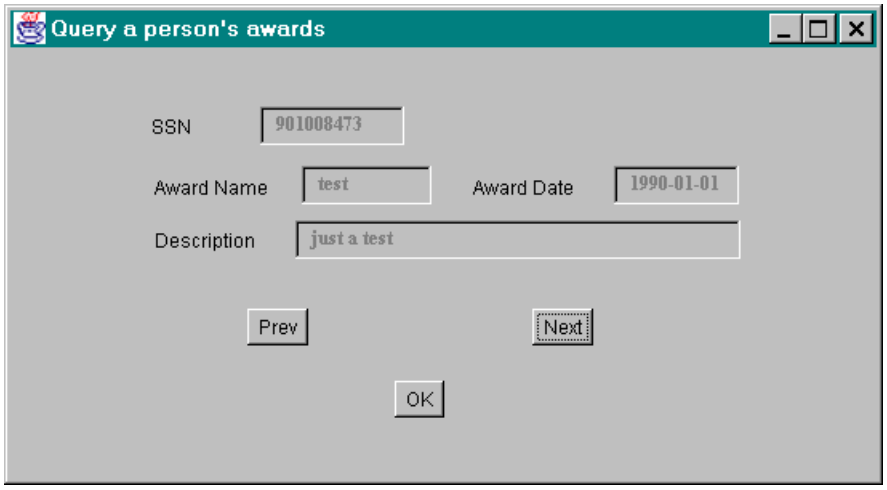
faculty -> admission -> decision

If you are the admission committee member, reenter your user id and password, then continue.



You can also modify or query the existing admission.

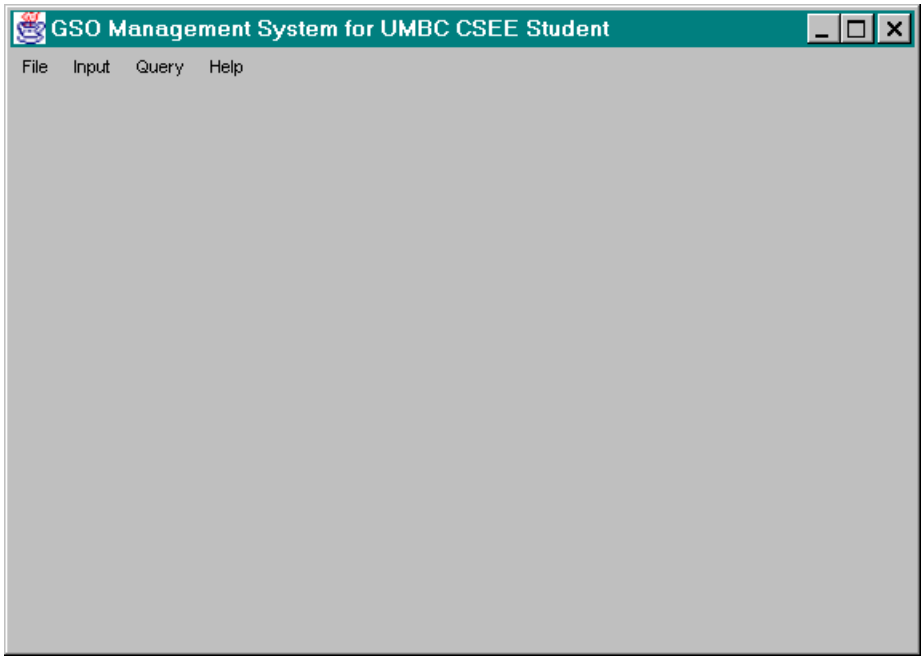
In academic affairs sections, faculty can query applicants' awards, publications, MS project, comp exams, Ph.D. preliminary and thesis.



In help section, it is the same as GSO staff, give help information and system version.

A5. Students

The system also give students an interface to input their own record and do query accordingly.



There are four major sections in this part.

File: exit the system safely.

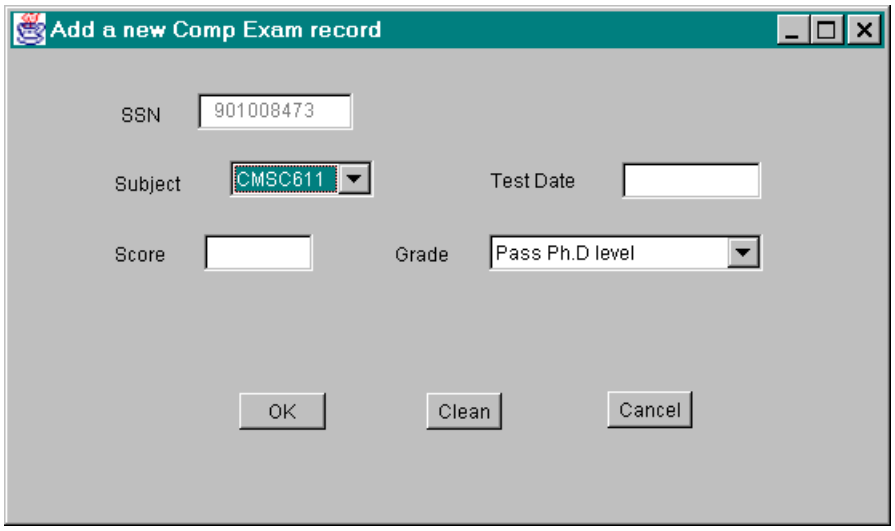
Input: Comp exams, Ph.D. preliminary, MS project, thesis, awards, and publications

Query: Comp exams, Ph.D. preliminary, MS project, thesis, awards, and publications

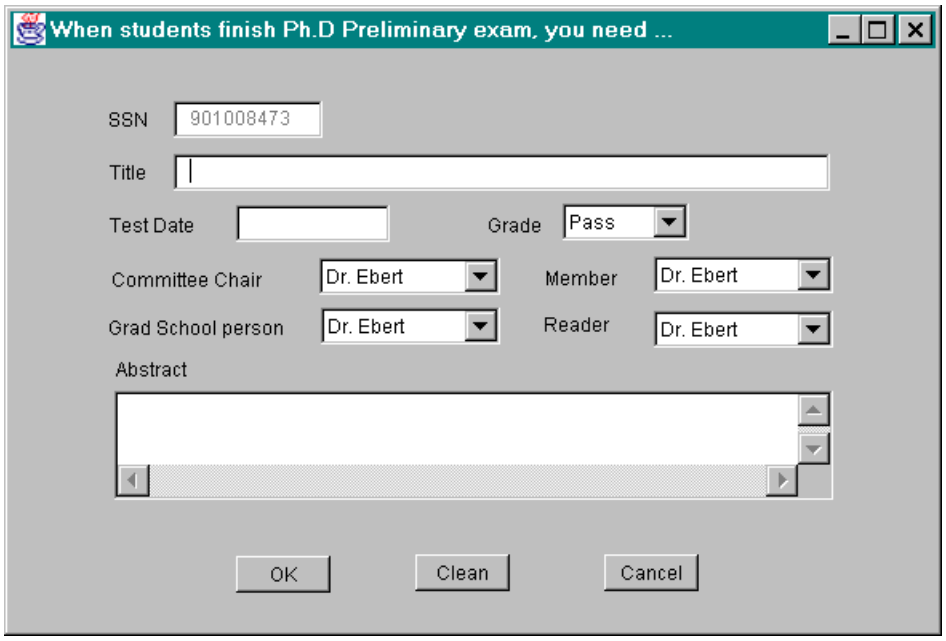
Help: manual and version.

A5.1 Input student information

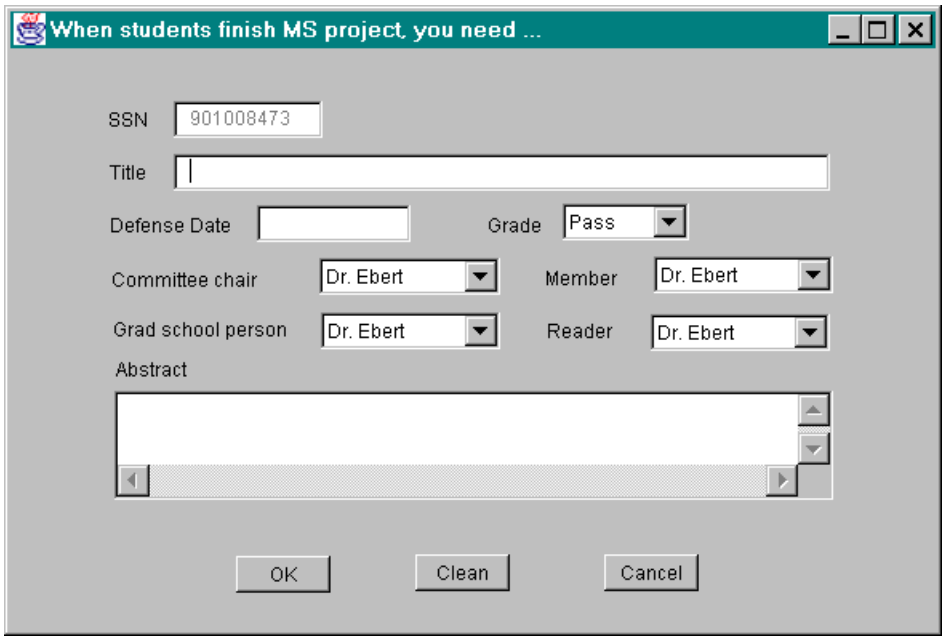
Comp exams: student -> input -> comp exam



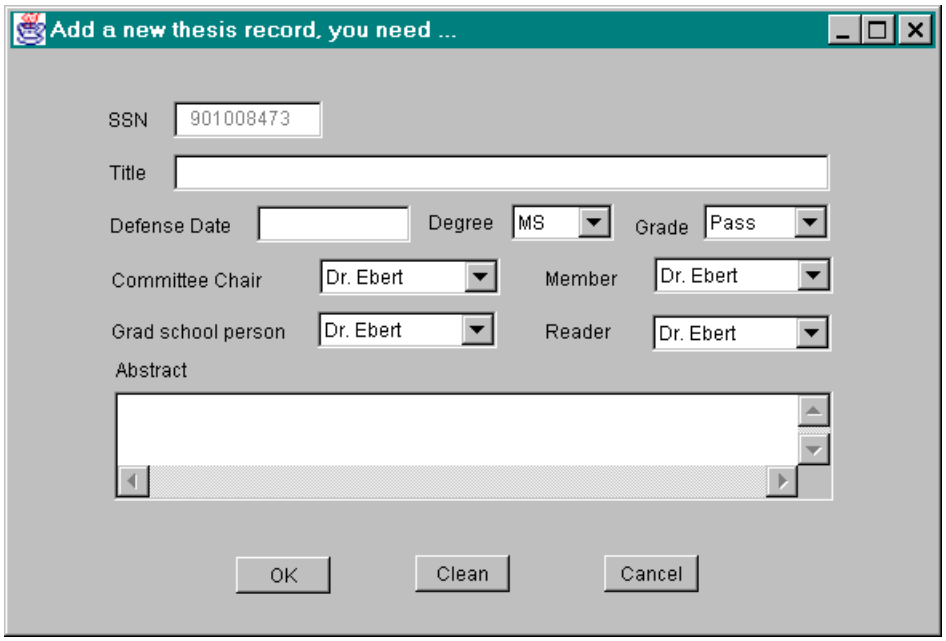
Ph.D. preliminary: student -> input -> Ph.D. preliminary



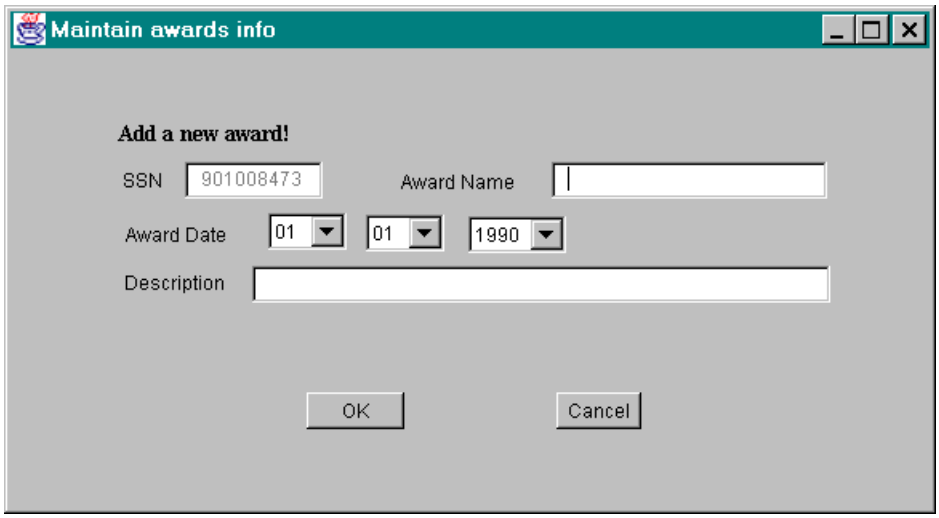
MS project: student -> input -> MS project



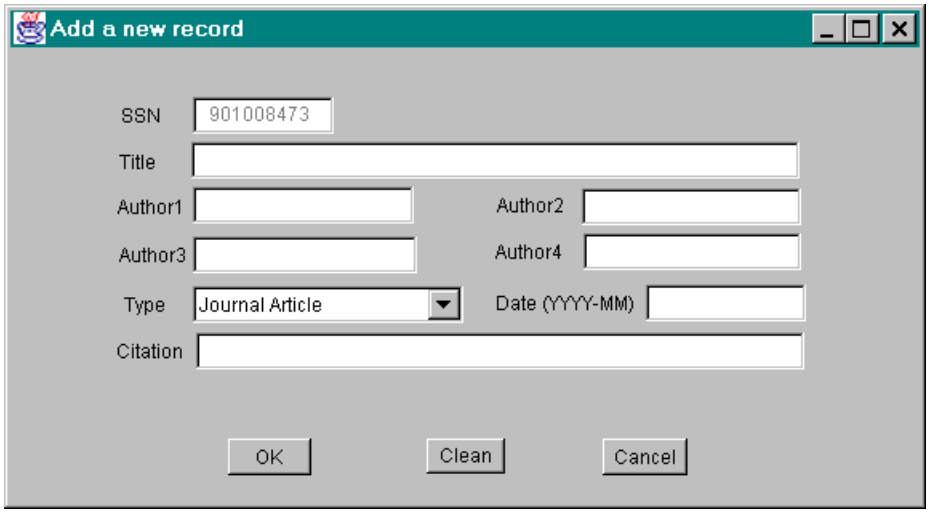
Thesis: student -> input -> thesis



Award: student -> input -> award



Publications: student -> input -> publications

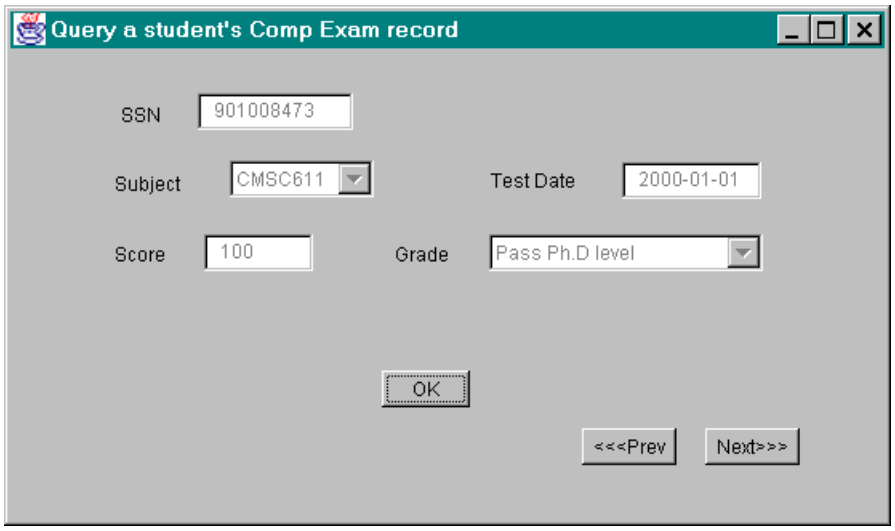


The students can only input their own information.

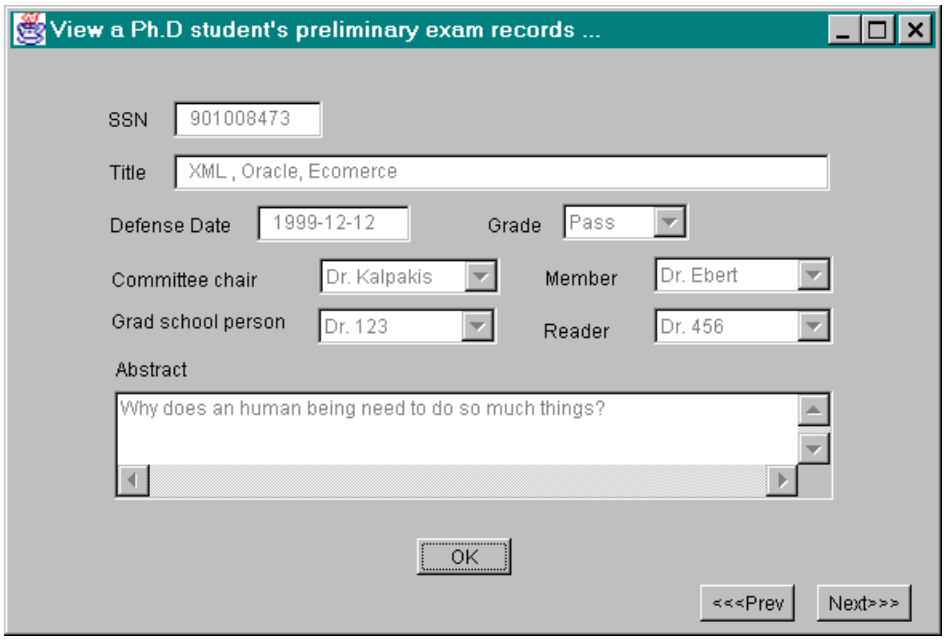
A5.2 Query student information

Students can also query the above information.

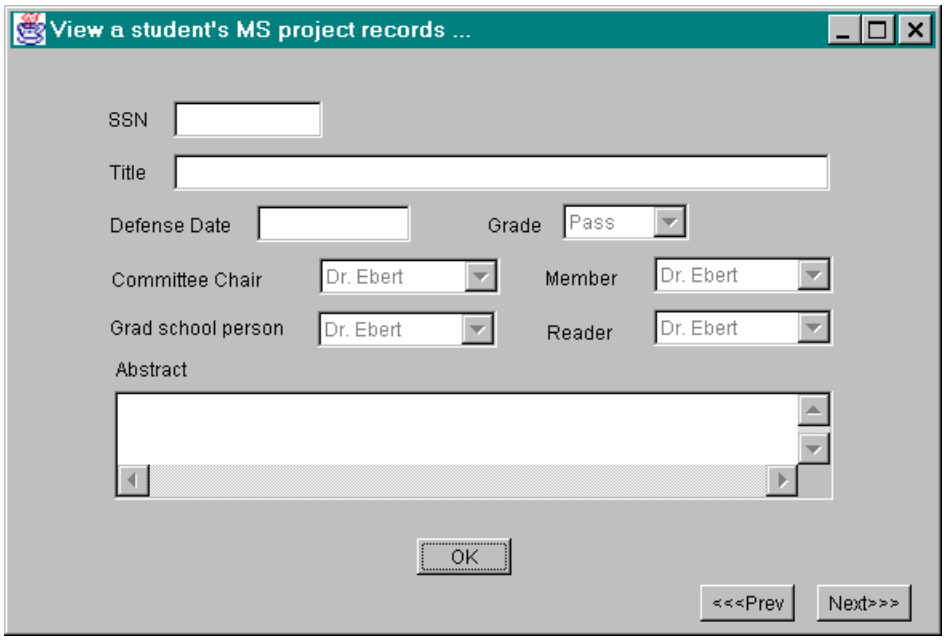
Comp exams: student -> Query -> Comp exam



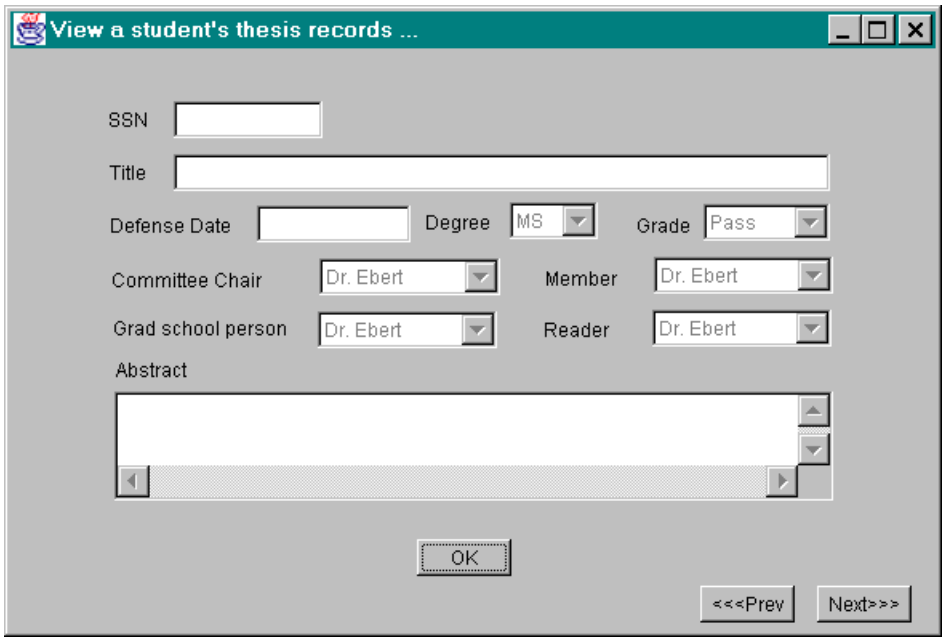
Ph.D. preliminary: student -> query -> Ph.D. preliminary



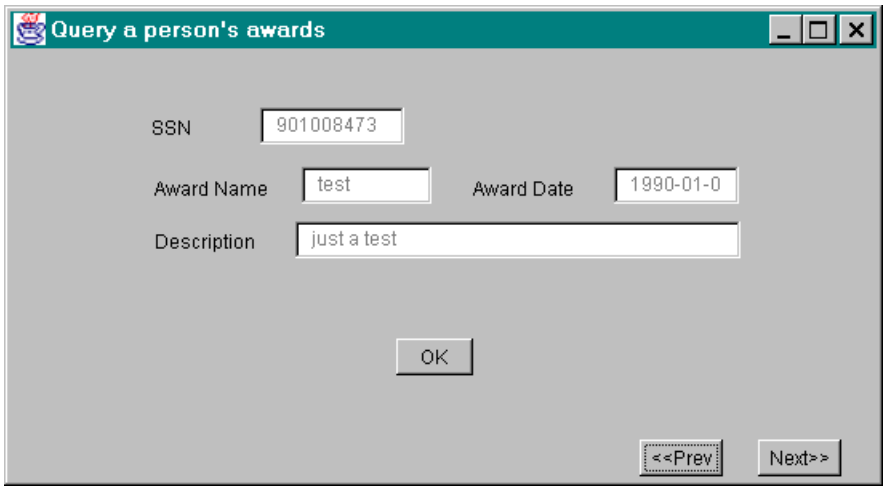
MS project: student -> query -> MS project



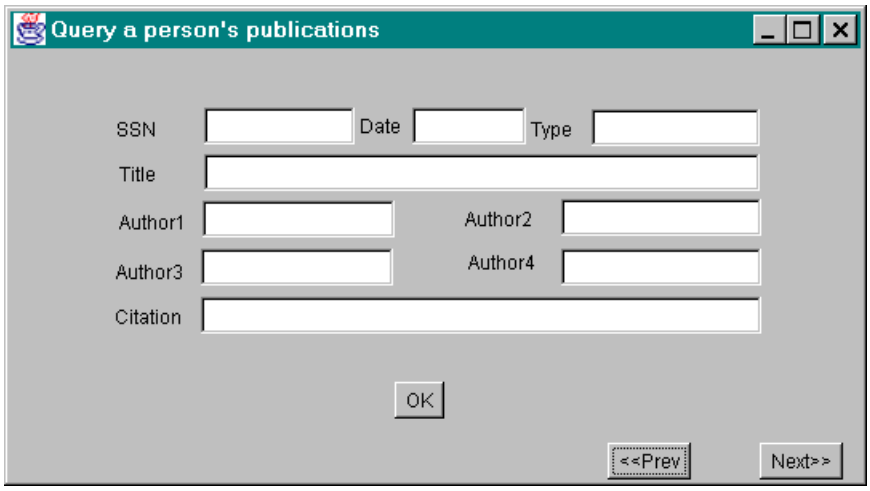
Thesis: student -> query -> thesis



Award: student -> query -> award



Publications: student -> query -> publication



A6. Other Information and Acknowledgement

SSN must be 9 digits without hyphen, if not, there will be a warning message prompted to you. Date type must use the format: YYYY-MM-DD

Also, for security reason, there are some limitations on student and faculty's operation. e.g. student can only input and view records of her/his-self. A faculty can only make a review on behalf of her/his-self.

The system has a friendly user interface, if there is any problem, a warning message would be prompted. If you come across any other problem, kindly please let us know:
Honglin Su, hosu@cs.umbc.edu
Shilang Tang, stang2@cs.umbc.edu
Zhongli Ding, zding1@cs.umbc.edu

Special thanks to Dr. Kalpakis, who instructed us during this semester, gave us a lot of help and encouragement.

Also thank LIST lab, we leveraged two NT machines for development.